- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields
# The Nebula Device 3 Data Structures

Here are the data structures with brief descriptions:

<table>
<thead>
<tr>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions::Action</td>
<td></td>
</tr>
<tr>
<td>Actions::ActionList</td>
<td></td>
</tr>
<tr>
<td>Actions::FSMAction</td>
<td></td>
</tr>
<tr>
<td>Actions::IfThenElseAction</td>
<td></td>
</tr>
<tr>
<td>Actions::SequenceAction</td>
<td></td>
</tr>
<tr>
<td>AngularPFeedbackLoop</td>
<td></td>
</tr>
<tr>
<td>Animation::AnimEventHandlerBase</td>
<td></td>
</tr>
<tr>
<td>Animation::AnimEventInfo</td>
<td></td>
</tr>
<tr>
<td>Animation::AnimEventServer</td>
<td></td>
</tr>
<tr>
<td>Animation::AnimJob</td>
<td></td>
</tr>
<tr>
<td>Animation::AnimJobEnqueueMode</td>
<td></td>
</tr>
<tr>
<td>Animation::AnimSequencer</td>
<td></td>
</tr>
<tr>
<td>Animation::PlayClipJob</td>
<td></td>
</tr>
<tr>
<td>Animator::AnimatorInstance</td>
<td></td>
</tr>
<tr>
<td>Animator::AnimLoopType</td>
<td></td>
</tr>
<tr>
<td>AnimatorNode</td>
<td></td>
</tr>
<tr>
<td>AnimKey</td>
<td></td>
</tr>
<tr>
<td>AnimKeyArray</td>
<td></td>
</tr>
<tr>
<td>App::Application</td>
<td></td>
</tr>
<tr>
<td>App::ConsoleApplication</td>
<td></td>
</tr>
<tr>
<td>App::GameApplication</td>
<td></td>
</tr>
<tr>
<td>App::RenderApplication</td>
<td></td>
</tr>
<tr>
<td>App::ViewerApplication</td>
<td></td>
</tr>
<tr>
<td>Application::StateHandler</td>
<td></td>
</tr>
<tr>
<td>Base::ArchiveFileSystemBase</td>
<td></td>
</tr>
<tr>
<td>Base::CalendarTimeBase</td>
<td></td>
</tr>
<tr>
<td>Base::DisplayDeviceBase</td>
<td></td>
</tr>
<tr>
<td>Base::GameContentServerBase</td>
<td></td>
</tr>
<tr>
<td>Base::GamePadBase</td>
<td></td>
</tr>
<tr>
<td>Base::IndexBufferBase</td>
<td></td>
</tr>
<tr>
<td>Base::InputServerBase</td>
<td></td>
</tr>
<tr>
<td>Base::JobBase</td>
<td></td>
</tr>
<tr>
<td>Base::JobFuncDescBase</td>
<td></td>
</tr>
<tr>
<td>Base::JobPortBase</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>Base::KeyboardBase</td>
<td></td>
</tr>
<tr>
<td>Base::MemoryIndexBufferLoaderBase</td>
<td></td>
</tr>
<tr>
<td>Base::MemoryVertexBufferLoaderBase</td>
<td></td>
</tr>
<tr>
<td>Base::MeshBase</td>
<td></td>
</tr>
<tr>
<td>Base::MouseBase</td>
<td></td>
</tr>
<tr>
<td>Base::MouseRenderDeviceBase</td>
<td></td>
</tr>
<tr>
<td>Base::MultipleRenderTargetBase</td>
<td></td>
</tr>
<tr>
<td>Base::ParticleRendererBase</td>
<td></td>
</tr>
<tr>
<td>Base::ParticleSystemInstanceBase</td>
<td></td>
</tr>
<tr>
<td>Base::RenderDeviceBase</td>
<td></td>
</tr>
<tr>
<td>Base::RenderTargetBase</td>
<td></td>
</tr>
<tr>
<td>Base::ResourceAllocator</td>
<td></td>
</tr>
<tr>
<td>Base::ResourceBase</td>
<td></td>
</tr>
<tr>
<td>Base::ResourceLump</td>
<td></td>
</tr>
<tr>
<td>Base::ShaderBase</td>
<td></td>
</tr>
<tr>
<td>Base::ShaderInstanceBase</td>
<td></td>
</tr>
<tr>
<td>Base::ShaderServerBase</td>
<td></td>
</tr>
<tr>
<td>Base::ShaderVariableBase</td>
<td></td>
</tr>
<tr>
<td>Base::ShaderVariableInstanceBase</td>
<td></td>
</tr>
<tr>
<td>Base::ShaderVariationBase</td>
<td></td>
</tr>
<tr>
<td>Base::ShapeRendererBase</td>
<td></td>
</tr>
<tr>
<td>Base::SkinnedMeshDrawInfoBase</td>
<td></td>
</tr>
<tr>
<td>Base::SkinnedMeshRendererBase</td>
<td></td>
</tr>
<tr>
<td>Base::StreamTextureSaverBase</td>
<td></td>
</tr>
<tr>
<td>Base::SystemInfoBase</td>
<td></td>
</tr>
<tr>
<td>Base::TextRendererBase</td>
<td></td>
</tr>
<tr>
<td>Base::TextureBase</td>
<td></td>
</tr>
<tr>
<td>Base::TextureBase::MapInfo</td>
<td>Access info filled by Map methods</td>
</tr>
<tr>
<td>Base::TransformDeviceBase</td>
<td></td>
</tr>
<tr>
<td>Base::VertexBufferBase</td>
<td></td>
</tr>
<tr>
<td>Base::VertexLayoutBase</td>
<td></td>
</tr>
<tr>
<td>Base::VertexLayoutServerBase</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::AnimEventManager</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::CategoryManager</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::CategoryManager::Category</td>
<td>Describes a category</td>
</tr>
<tr>
<td>BaseGameFeature::CategoryManager::Entry</td>
<td>Describes an entry in a value table</td>
</tr>
<tr>
<td>BaseGameFeature::EntityLoader</td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>BaseGameFeature::EntityLoaderBase</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::EntityManager</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::EntityManager::DelayedJob</td>
<td>Delayed jobs</td>
</tr>
<tr>
<td>BaseGameFeature::EnvEntityManager</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::EnvironmentLoader</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::EnvQueryManager</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::FactoryManager</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::FactoryManager::PropertyEntry</td>
<td>Entity blueprint, these are created by ParseBlueprints()</td>
</tr>
<tr>
<td>BaseGameFeature::FocusManager</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::GameStateHandler</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::GlobalAttrsManager</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::InputTimeSource</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::LevelLoader</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::LoaderServer</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::SystemTimeSource</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::TimeManager</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::TimeProperty</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::TimeSource</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::TimingTimeSource</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::TransformableProperty</td>
<td></td>
</tr>
<tr>
<td>BaseGameFeature::UserProfile</td>
<td></td>
</tr>
<tr>
<td>Characters::Character</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterAnimationController</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterInstance</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterJoint</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterNode</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterNodeInstance</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterServer</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterSkeleton</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterSkeletonInstance</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterSkin</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterSkinLibrary</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterSkinList</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterSkinNode</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterSkinNodeInstance</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterSkinSet</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterSkinSetInstance</td>
<td></td>
</tr>
<tr>
<td>Characters::CharacterSkinVariationSet</td>
<td></td>
</tr>
<tr>
<td>Characters::CharJointComponents</td>
<td></td>
</tr>
<tr>
<td><strong>Total Classes</strong>: 48</td>
<td></td>
</tr>
<tr>
<td>Characters::SkinnedMeshDrawInfo</td>
<td>Characters::SkinnedMeshRenderer</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>CoreAnimation::AnimClip</td>
<td>CoreAnimation::AnimCurve</td>
</tr>
<tr>
<td>Class Name</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::DisplayDevice</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::DisplayEvent</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::DisplayEventHandler</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::DisplayMode</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::ImageFileFormat</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::IndexBuffer</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::IndexType</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::MemoryIndexBufferLoader</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::MemoryMeshLoader</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::MemoryVertexBufferLoader</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::Mesh</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::MousePointer</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::MouseRenderDevice</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::MultipleRenderTarget</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::PixelFormat</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::PrimitiveGroup</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::PrimitiveTopology</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::RenderDevice</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::RenderEvent</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::RenderEventHandler</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::RenderShape</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::RenderTarget</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::Shader</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::ShaderFeature</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::ShaderInstance</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::ShaderServer</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::ShaderVariable</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::ShaderVariableInstance</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::ShapeRenderer</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::StreamMeshLoader</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::StreamShaderLoader</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::StreamTextureLoader</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::StreamTextureSaver</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::TextElement</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::TextRenderer</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::Texture</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::ThreadSafeDisplayEventHandler</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td></td>
</tr>
<tr>
<td>------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::ThreadSafeRenderEventHandler</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::TransformDevice</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::VertexBuffer</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::VertexComponent</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::VertexLayout</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::VertexLayoutServer</td>
<td></td>
</tr>
<tr>
<td>Debug::ConsolePageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::CorePageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::DebugCounter</td>
<td></td>
</tr>
<tr>
<td>Debug::DebugFloat</td>
<td></td>
</tr>
<tr>
<td>Debug::DebugGraphicsHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::DebugHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::DebugInterface</td>
<td></td>
</tr>
<tr>
<td>Debug::DebugPageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::DebugServer</td>
<td></td>
</tr>
<tr>
<td>Debug::DebugShapeRenderer</td>
<td></td>
</tr>
<tr>
<td>Debug::DebugTextRenderer</td>
<td></td>
</tr>
<tr>
<td>Debug::DebugTimer</td>
<td></td>
</tr>
<tr>
<td>Debug::DisplayPageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::GraphicsPageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::HelloWorldRequestHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::IoPageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::MemoryPageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::MeshPageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::MiniDump</td>
<td></td>
</tr>
<tr>
<td>Debug::ObjectInspectorHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::ShaderPageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::StreamingTexturePageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::SvgTestPageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::TexturePageHandler</td>
<td></td>
</tr>
<tr>
<td>Debug::ThreadPageHandler</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9DisplayDevice</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9ParticleRenderer</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9ParticleSystemInstance</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9RenderDevice</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9RenderTarget</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9Shader</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9ShaderInstance</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9ShaderServer</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9ShaderVariable</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9ShaderVariation</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9StreamShaderLoader</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9StreamTextureLoader</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9TextRenderer</td>
<td></td>
</tr>
<tr>
<td>Direct3D9::D3D9Texture</td>
<td></td>
</tr>
<tr>
<td>Frame::FrameBatch</td>
<td></td>
</tr>
<tr>
<td>Frame::FramePass</td>
<td></td>
</tr>
<tr>
<td>Frame::FramePassBase</td>
<td></td>
</tr>
<tr>
<td>Frame::FramePostEffect</td>
<td></td>
</tr>
<tr>
<td>Frame::FrameServer</td>
<td></td>
</tr>
<tr>
<td>Frame::FrameShader</td>
<td></td>
</tr>
<tr>
<td>Frame::FrameShaderLoader</td>
<td></td>
</tr>
<tr>
<td>Frame::LightingMode</td>
<td></td>
</tr>
<tr>
<td>Frame::SortingMode</td>
<td></td>
</tr>
<tr>
<td>FrameSync::FrameSyncHandlerThread</td>
<td></td>
</tr>
<tr>
<td>FrameSync::FrameSyncSharedData</td>
<td></td>
</tr>
<tr>
<td>FrameSync::FrameSyncTimer</td>
<td></td>
</tr>
<tr>
<td>FSM::State</td>
<td></td>
</tr>
<tr>
<td>FSM::StateMachine</td>
<td></td>
</tr>
<tr>
<td>FSM::Transition</td>
<td></td>
</tr>
<tr>
<td>Game::BaseGameFeatureUnit</td>
<td></td>
</tr>
<tr>
<td>Game::Entity</td>
<td></td>
</tr>
<tr>
<td>Game::FeatureUnit</td>
<td></td>
</tr>
<tr>
<td>Game::GameServer</td>
<td></td>
</tr>
<tr>
<td>Game::Manager</td>
<td></td>
</tr>
<tr>
<td>Game::PhysicsFeatureUnit</td>
<td></td>
</tr>
<tr>
<td>Game::Property</td>
<td></td>
</tr>
<tr>
<td>Graphics::AbstractLightEntity</td>
<td></td>
</tr>
<tr>
<td>Graphics::CameraEntity</td>
<td></td>
</tr>
<tr>
<td>Graphics::Display</td>
<td></td>
</tr>
<tr>
<td>Graphics::DisplaySettings</td>
<td></td>
</tr>
<tr>
<td>Graphics::GlobalLightEntity</td>
<td></td>
</tr>
<tr>
<td>Graphics::GraphicsEntity</td>
<td></td>
</tr>
<tr>
<td>Graphics::GraphicsEntityType</td>
<td></td>
</tr>
<tr>
<td>Graphics::GraphicsHandler</td>
<td></td>
</tr>
<tr>
<td>Graphics::GraphicsInterface</td>
<td></td>
</tr>
<tr>
<td>Graphics::GraphicsServer</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td></td>
</tr>
<tr>
<td>------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Graphics::ModelEntity</td>
<td></td>
</tr>
<tr>
<td>Graphics::MouseRenderer</td>
<td></td>
</tr>
<tr>
<td>Graphics::PointLightEntity</td>
<td></td>
</tr>
<tr>
<td>Graphics::SpotLightEntity</td>
<td></td>
</tr>
<tr>
<td>Graphics::View</td>
<td></td>
</tr>
<tr>
<td>GraphicsFeature::ActorGraphicsProperty</td>
<td></td>
</tr>
<tr>
<td>GraphicsFeature::AnimationControlProperty</td>
<td></td>
</tr>
<tr>
<td>GraphicsFeature::AttachmentManager</td>
<td></td>
</tr>
<tr>
<td>GraphicsFeature::CameraProperty</td>
<td></td>
</tr>
<tr>
<td>GraphicsFeature::ChaseCameraProperty</td>
<td></td>
</tr>
<tr>
<td>GraphicsFeature::GraphicsFeatureUnitUnit</td>
<td></td>
</tr>
<tr>
<td>GraphicsFeature::GraphicsProperty</td>
<td></td>
</tr>
<tr>
<td>GraphicsFeature::LightFlickerUtil</td>
<td></td>
</tr>
<tr>
<td>GraphicsFeature::MayaCameraProperty</td>
<td></td>
</tr>
<tr>
<td>GraphicsFeature::SegmentedGfxUtil</td>
<td></td>
</tr>
<tr>
<td>Http::DefaultHttpRequestHandler</td>
<td></td>
</tr>
<tr>
<td>Http::HtmlElement</td>
<td></td>
</tr>
<tr>
<td>Http::HtmlPageWriter</td>
<td></td>
</tr>
<tr>
<td>Http::HttpClient</td>
<td></td>
</tr>
<tr>
<td>Http::HttpClientRegistry</td>
<td></td>
</tr>
<tr>
<td>Http::HttpInterface</td>
<td></td>
</tr>
<tr>
<td>Http::HttpMessageHandler</td>
<td></td>
</tr>
<tr>
<td>Http::HttpMethod</td>
<td></td>
</tr>
<tr>
<td>Http::Http NzStream</td>
<td></td>
</tr>
<tr>
<td>Http::HttpRequest</td>
<td></td>
</tr>
<tr>
<td>Http::HttpRequestHandler</td>
<td></td>
</tr>
<tr>
<td>Http::HttpRequestReader</td>
<td></td>
</tr>
<tr>
<td>Http::HttpRequestWriter</td>
<td></td>
</tr>
<tr>
<td>Http::HttpResponseReader</td>
<td></td>
</tr>
<tr>
<td>Http::HttpResponseWriter</td>
<td></td>
</tr>
<tr>
<td>Http::HttpServer</td>
<td></td>
</tr>
<tr>
<td>Http::HttpServerProxy</td>
<td></td>
</tr>
<tr>
<td>Http::HttpStatus</td>
<td></td>
</tr>
<tr>
<td>Http::HttpStream</td>
<td></td>
</tr>
<tr>
<td>Http::SvgLineChartWriter</td>
<td></td>
</tr>
<tr>
<td>Http::SvgPageWriter</td>
<td></td>
</tr>
<tr>
<td>Input::GamePad</td>
<td></td>
</tr>
<tr>
<td>Input::InputEvent</td>
<td></td>
</tr>
<tr>
<td>Input::InputHandler</td>
<td></td>
</tr>
<tr>
<td>Input::InputPriority</td>
<td></td>
</tr>
<tr>
<td>Input::InputServer</td>
<td></td>
</tr>
<tr>
<td>Input::Key</td>
<td></td>
</tr>
<tr>
<td>Input::Keyboard</td>
<td></td>
</tr>
<tr>
<td>Input::Mouse</td>
<td></td>
</tr>
<tr>
<td>Input::MouseButton</td>
<td></td>
</tr>
<tr>
<td>Interface::InterfaceBase</td>
<td></td>
</tr>
<tr>
<td>Interface::InterfaceHandlerBase</td>
<td></td>
</tr>
<tr>
<td>InternalGraphics::AttachmentServer</td>
<td></td>
</tr>
<tr>
<td>InternalGraphics::InternalCameraEntity</td>
<td></td>
</tr>
<tr>
<td>InternalGraphics::InternalGraphicsEntity</td>
<td></td>
</tr>
<tr>
<td>InternalGraphics::InternalGraphicsEntityGenericType</td>
<td></td>
</tr>
<tr>
<td>InternalGraphics::InternalGraphicsServer</td>
<td></td>
</tr>
<tr>
<td>InternalGraphics::InternalModelEntity</td>
<td></td>
</tr>
<tr>
<td>InternalGraphics::InternalStage</td>
<td></td>
</tr>
<tr>
<td>InternalGraphics::InternalView</td>
<td></td>
</tr>
<tr>
<td>InternalGraphics::VisibilityCell</td>
<td></td>
</tr>
<tr>
<td>IO::Archive</td>
<td></td>
</tr>
<tr>
<td>IO::ArchiveBase</td>
<td></td>
</tr>
<tr>
<td>IO::ArchiveFileSystem</td>
<td></td>
</tr>
<tr>
<td>IO::Assign</td>
<td></td>
</tr>
<tr>
<td>IO::AssignRegistry</td>
<td></td>
</tr>
<tr>
<td>IO::BinaryReader</td>
<td></td>
</tr>
<tr>
<td>IO::BinaryWriter</td>
<td></td>
</tr>
<tr>
<td>IO::BXmlLoaderUtil</td>
<td></td>
</tr>
<tr>
<td>IO::BXmlReader</td>
<td></td>
</tr>
<tr>
<td>IO::Console</td>
<td></td>
</tr>
<tr>
<td>IO::ConsoleHandler</td>
<td></td>
</tr>
<tr>
<td>IO::ExcelXmlReader</td>
<td></td>
</tr>
<tr>
<td>IO::FileStream</td>
<td></td>
</tr>
<tr>
<td>IO::FileTime</td>
<td></td>
</tr>
<tr>
<td>IO::FSWrapper</td>
<td></td>
</tr>
<tr>
<td>IO::GameContentServer</td>
<td></td>
</tr>
<tr>
<td>IO::HistoryConsoleHandler</td>
<td></td>
</tr>
<tr>
<td>IO::Interface</td>
<td></td>
</tr>
<tr>
<td>IO::IoInterfaceHandler</td>
<td></td>
</tr>
<tr>
<td>IO::IoServer</td>
<td></td>
</tr>
<tr>
<td>IO::LogFileConsoleHandler</td>
<td></td>
</tr>
<tr>
<td>IO::MediaType</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Class Name</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>IO</td>
<td>IO::MemoryStream</td>
</tr>
<tr>
<td></td>
<td>IO::SchemeRegistry</td>
</tr>
<tr>
<td></td>
<td>IO::Stream</td>
</tr>
<tr>
<td></td>
<td>IO::StreamReader</td>
</tr>
<tr>
<td></td>
<td>IO::StreamWriter</td>
</tr>
<tr>
<td></td>
<td>IO::TextWriter</td>
</tr>
<tr>
<td></td>
<td>IO::URI</td>
</tr>
<tr>
<td></td>
<td>IO::XmlReader</td>
</tr>
<tr>
<td></td>
<td>IO::XmlWriter</td>
</tr>
<tr>
<td></td>
<td>IO::ZipArchive</td>
</tr>
<tr>
<td></td>
<td>IO::ZipDirEntry</td>
</tr>
<tr>
<td></td>
<td>IO::ZipFileEntry</td>
</tr>
<tr>
<td></td>
<td>IO::ZipFileStream</td>
</tr>
<tr>
<td></td>
<td>IO::ZipFileSystem</td>
</tr>
<tr>
<td>Jobs</td>
<td>Jobs::Job</td>
</tr>
<tr>
<td></td>
<td>Jobs::JobContext</td>
</tr>
<tr>
<td></td>
<td>Jobs::JobDataDesc</td>
</tr>
<tr>
<td></td>
<td>Jobs::JobFuncDesc</td>
</tr>
<tr>
<td></td>
<td>Jobs::JobPort</td>
</tr>
<tr>
<td></td>
<td>Jobs::JobServerBase</td>
</tr>
<tr>
<td></td>
<td>Jobs::JobSystem</td>
</tr>
<tr>
<td></td>
<td>Jobs::JobUniformDesc</td>
</tr>
<tr>
<td></td>
<td>Jobs::SerialJob</td>
</tr>
<tr>
<td></td>
<td>Jobs::SerialJobFuncDesc</td>
</tr>
<tr>
<td></td>
<td>Jobs::SerialJobPort</td>
</tr>
<tr>
<td></td>
<td>Jobs::SerialJobSystem</td>
</tr>
<tr>
<td></td>
<td>Jobs::TPJob</td>
</tr>
<tr>
<td></td>
<td>Jobs::TPJobCommand</td>
</tr>
<tr>
<td></td>
<td>Jobs::TPJobFuncDesc</td>
</tr>
<tr>
<td></td>
<td>Jobs::TPJobPort</td>
</tr>
<tr>
<td></td>
<td>Jobs::TPJobSlice</td>
</tr>
<tr>
<td></td>
<td>Jobs::TPJobSystem</td>
</tr>
<tr>
<td></td>
<td>Jobs::TPJobThreadPool</td>
</tr>
<tr>
<td></td>
<td>Jobs::TPWorkerThread</td>
</tr>
<tr>
<td>Legacy</td>
<td>Legacy::Nvx2StreamReader</td>
</tr>
<tr>
<td>Lighting</td>
<td>Lighting::InternalAbstractLightEntity</td>
</tr>
<tr>
<td></td>
<td>Lighting::InternalGlobalLightEntity</td>
</tr>
<tr>
<td></td>
<td>Lighting::InternalPointLightEntity</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td><code>Lighting::InternalSpotLightEntity</code></td>
<td></td>
</tr>
<tr>
<td><code>Lighting::LightPrePassServer</code></td>
<td></td>
</tr>
<tr>
<td><code>Lighting::LightServer</code></td>
<td></td>
</tr>
<tr>
<td><code>Lighting::LightServerBase</code></td>
<td></td>
</tr>
<tr>
<td><code>Lighting::LightType</code></td>
<td></td>
</tr>
<tr>
<td><code>Lighting::PSSMUtil</code></td>
<td></td>
</tr>
<tr>
<td><code>Lighting::ShadowServer</code></td>
<td></td>
</tr>
<tr>
<td><code>Lighting::SM30LightServer</code></td>
<td></td>
</tr>
<tr>
<td><code>Lighting::SM30ShadowServer</code></td>
<td></td>
</tr>
<tr>
<td><code>Managers::LogManager</code></td>
<td></td>
</tr>
<tr>
<td><code>Managers::QuestManager</code></td>
<td></td>
</tr>
<tr>
<td><code>Managers::ScriptTable</code></td>
<td></td>
</tr>
<tr>
<td><code>Managers::ScriptTemplateManager</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::bbox</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::ClipStatus</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::Extrapolator&lt; TYPE &gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::float2</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::float4</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::frustum</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::line</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::matrix44</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::noise</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::plane</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::point</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::polar</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::quaternion</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::rectangle&lt; TYPE &gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::sphere</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::transform44</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::vector</code></td>
<td></td>
</tr>
<tr>
<td><code>Memory::Heap</code></td>
<td></td>
</tr>
<tr>
<td><code>Memory::Memory</code></td>
<td></td>
</tr>
<tr>
<td><code>Memory::MemoryPool</code></td>
<td></td>
</tr>
<tr>
<td><code>Memory::PoolArrayAllocator</code></td>
<td></td>
</tr>
<tr>
<td><code>Memory::TotalMemoryStatus</code></td>
<td></td>
</tr>
<tr>
<td><code>Messaging::AsyncPort</code></td>
<td></td>
</tr>
<tr>
<td><code>Messaging::BatchMessage</code></td>
<td></td>
</tr>
<tr>
<td><code>Messaging::BlockingHandlerThread</code></td>
<td></td>
</tr>
<tr>
<td><code>Messaging::DelegateTable</code></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Messaging::Dispatcher</td>
<td></td>
</tr>
<tr>
<td>Messaging::Handler</td>
<td></td>
</tr>
<tr>
<td>Messaging::HandlerThreadBase</td>
<td></td>
</tr>
<tr>
<td>Messaging::Id</td>
<td></td>
</tr>
<tr>
<td>Messaging::Message</td>
<td></td>
</tr>
<tr>
<td>Messaging::MessageReader</td>
<td></td>
</tr>
<tr>
<td>Messaging::MessageWriter</td>
<td></td>
</tr>
<tr>
<td>Messaging::Port</td>
<td></td>
</tr>
<tr>
<td>Messaging::RunThroughHandlerThread</td>
<td></td>
</tr>
<tr>
<td>Messaging::StaticMessageHandler</td>
<td></td>
</tr>
<tr>
<td>MGraphics::VisibilityBox</td>
<td></td>
</tr>
<tr>
<td>MGraphics::VisibilityCluster</td>
<td></td>
</tr>
<tr>
<td>Models::ManagedModel</td>
<td></td>
</tr>
<tr>
<td>Models::Model</td>
<td></td>
</tr>
<tr>
<td>Models::ModelInstance</td>
<td></td>
</tr>
<tr>
<td>Models::ModelNode</td>
<td></td>
</tr>
<tr>
<td>Models::ModelNodeInstance</td>
<td></td>
</tr>
<tr>
<td>Models::ModelNodeType</td>
<td></td>
</tr>
<tr>
<td>Models::ModelServer</td>
<td></td>
</tr>
<tr>
<td>Models::ShapeNode</td>
<td></td>
</tr>
<tr>
<td>Models::ShapeNodeInstance</td>
<td></td>
</tr>
<tr>
<td>Models::StateNode</td>
<td></td>
</tr>
<tr>
<td>Models::StateNodeInstance</td>
<td></td>
</tr>
<tr>
<td>Models::StreamModelLoader</td>
<td></td>
</tr>
<tr>
<td>Models::TransformNode</td>
<td></td>
</tr>
<tr>
<td>Models::TransformNodeInstance</td>
<td></td>
</tr>
<tr>
<td>Models::VisResolveContainer&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Models::VisResolver</td>
<td></td>
</tr>
<tr>
<td>Net::DebugMessage</td>
<td></td>
</tr>
<tr>
<td>Net::DebugPacket</td>
<td></td>
</tr>
<tr>
<td>Net::IpAddress</td>
<td></td>
</tr>
<tr>
<td>Net::MessageClient</td>
<td></td>
</tr>
<tr>
<td>Net::MessageClientConnection</td>
<td></td>
</tr>
<tr>
<td>Net::Socket</td>
<td></td>
</tr>
<tr>
<td>Net::StdTcpClient</td>
<td></td>
</tr>
<tr>
<td>Net::StdTcpClientConnection</td>
<td></td>
</tr>
<tr>
<td>Net::StdTcpServer</td>
<td></td>
</tr>
<tr>
<td>Net::TcpClient</td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Net::TcpClientConnection</td>
<td></td>
</tr>
<tr>
<td>Net::TcpMessageCodec</td>
<td></td>
</tr>
<tr>
<td>Net::TcpServer</td>
<td></td>
</tr>
<tr>
<td>OSX::OSXConsoleHandler</td>
<td></td>
</tr>
<tr>
<td>OSX::OSXCpu</td>
<td></td>
</tr>
<tr>
<td>OSX::OSXCriticalSection</td>
<td></td>
</tr>
<tr>
<td>OSX::OSXFileTime</td>
<td></td>
</tr>
<tr>
<td>OSX::OSXFSWrapper</td>
<td></td>
</tr>
<tr>
<td>OSX::OSXGuid</td>
<td></td>
</tr>
<tr>
<td>OSX::OSXHeap</td>
<td></td>
</tr>
<tr>
<td>OSX::OSXInterlocked</td>
<td></td>
</tr>
<tr>
<td>OSX::OSXMemoryPool</td>
<td></td>
</tr>
<tr>
<td>OSX::OSXThreadId</td>
<td></td>
</tr>
<tr>
<td>OSX::OSXThreadIdLocalPtr&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>OSX::SysFunc</td>
<td></td>
</tr>
<tr>
<td>Particles::EmitterAttrs</td>
<td></td>
</tr>
<tr>
<td>Particles::EmitterMesh</td>
<td></td>
</tr>
<tr>
<td>Particles::EnvelopeCurve</td>
<td></td>
</tr>
<tr>
<td>Particles::EnvelopeSampleBuffer</td>
<td></td>
</tr>
<tr>
<td>Particles::Particle</td>
<td></td>
</tr>
<tr>
<td>Particles::ParticleRenderer</td>
<td></td>
</tr>
<tr>
<td>Particles::ParticleRenderInfo</td>
<td></td>
</tr>
<tr>
<td>Particles::ParticleServer</td>
<td></td>
</tr>
<tr>
<td>Particles::ParticleSystem</td>
<td></td>
</tr>
<tr>
<td>Particles::ParticleSystemNode</td>
<td></td>
</tr>
<tr>
<td>Particles::ParticleSystemNodeInstance</td>
<td></td>
</tr>
<tr>
<td>Particles::ParticleSystemRenderer</td>
<td></td>
</tr>
<tr>
<td>Particles::ParticleSystemState</td>
<td></td>
</tr>
<tr>
<td>PFeedbackLoop</td>
<td></td>
</tr>
<tr>
<td>PhysicsFeature::ActorPhysicsProperty</td>
<td></td>
</tr>
<tr>
<td>PhysicsFeature::EnvironmentCollideProperty</td>
<td></td>
</tr>
<tr>
<td>PhysicsFeature::MouseGripperProperty</td>
<td></td>
</tr>
<tr>
<td>PhysicsFeature::PhysicsProperty</td>
<td></td>
</tr>
<tr>
<td>PhysicsFeature::TriggerProperty</td>
<td></td>
</tr>
<tr>
<td>PIDFeedbackLoop</td>
<td></td>
</tr>
<tr>
<td>PriorityArray</td>
<td></td>
</tr>
<tr>
<td>Properties::AmbienceBubbleProperty</td>
<td></td>
</tr>
<tr>
<td>Class/Module</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>Properties::BehaviourProperty</td>
<td></td>
</tr>
<tr>
<td>Properties::CutsceneCameraProperty</td>
<td></td>
</tr>
<tr>
<td>Properties::EnvironmentGraphicsProperty</td>
<td></td>
</tr>
<tr>
<td>Properties::InputProperty</td>
<td></td>
</tr>
<tr>
<td>Properties::LightProperty</td>
<td></td>
</tr>
<tr>
<td>Properties::PathAnimProperty</td>
<td></td>
</tr>
<tr>
<td>Ptr&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>QuadTree</td>
<td></td>
</tr>
<tr>
<td>Render::RenderConsoleHandler</td>
<td></td>
</tr>
<tr>
<td>RenderModules::RenderModule</td>
<td></td>
</tr>
<tr>
<td>RenderModules::RTPlugin</td>
<td></td>
</tr>
<tr>
<td>RenderModules::RTPluginRegistry</td>
<td></td>
</tr>
<tr>
<td>RenderUtil::DrawFullScreenQuad</td>
<td></td>
</tr>
<tr>
<td>RenderUtil::MayaCameraUtil</td>
<td></td>
</tr>
<tr>
<td>RenderUtil::MouseRayUtil</td>
<td></td>
</tr>
<tr>
<td>RenderUtil::NodeLookupUtil</td>
<td></td>
</tr>
<tr>
<td>Resources::D3D9TextureStreamer</td>
<td></td>
</tr>
<tr>
<td>Resources::LoadingResource</td>
<td></td>
</tr>
<tr>
<td>Resources::ManagedMesh</td>
<td></td>
</tr>
<tr>
<td>Resources::ManagedResource</td>
<td></td>
</tr>
<tr>
<td>Resources::ManagedTexture</td>
<td></td>
</tr>
<tr>
<td>Resources::PoolLoadingResource</td>
<td></td>
</tr>
<tr>
<td>Resources::PoolResourceMapper</td>
<td></td>
</tr>
<tr>
<td>Resources::PoolSetupInfo</td>
<td>Helper class for pool generation</td>
</tr>
<tr>
<td>Resources::Resource</td>
<td></td>
</tr>
<tr>
<td>Resources::ResourceDictionary</td>
<td></td>
</tr>
<tr>
<td>Resources::ResourceDictionary::Entry</td>
<td>Resource dictionary entry</td>
</tr>
<tr>
<td>Resources::ResourceId</td>
<td></td>
</tr>
<tr>
<td>Resources::ResourceLoader</td>
<td></td>
</tr>
<tr>
<td>Resources::ResourceManager</td>
<td></td>
</tr>
<tr>
<td>Resources::ResourceMapper</td>
<td></td>
</tr>
<tr>
<td>Resources::ResourceSaver</td>
<td></td>
</tr>
<tr>
<td>Resources::ResourceScheduler</td>
<td></td>
</tr>
<tr>
<td>Resources::SimpleResourceMapper</td>
<td></td>
</tr>
<tr>
<td>Resources::StreamResourceLoader</td>
<td></td>
</tr>
<tr>
<td>Resources::TexturePoolMapperScheduler</td>
<td></td>
</tr>
<tr>
<td>Resources::TextureStreamer</td>
<td></td>
</tr>
<tr>
<td>Round</td>
<td></td>
</tr>
<tr>
<td>Class Path</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Script::ActionReader</td>
<td></td>
</tr>
<tr>
<td>Script::Attributes</td>
<td></td>
</tr>
<tr>
<td>Script::Dialog</td>
<td></td>
</tr>
<tr>
<td>Script::DialogDesc</td>
<td></td>
</tr>
<tr>
<td>Script::DialogManager</td>
<td></td>
</tr>
<tr>
<td>Script::DialogTake</td>
<td></td>
</tr>
<tr>
<td>Script::InfoLog</td>
<td></td>
</tr>
<tr>
<td>Script::InfoLogFilter</td>
<td></td>
</tr>
<tr>
<td>Script::ScriptManager</td>
<td></td>
</tr>
<tr>
<td>Script::SubstitutionManager</td>
<td></td>
</tr>
<tr>
<td>Script::Task</td>
<td></td>
</tr>
<tr>
<td>ScriptFeature::ScriptFeatureUnit</td>
<td></td>
</tr>
<tr>
<td>Shared::CameraSettings</td>
<td></td>
</tr>
<tr>
<td>Shared::CharJointInfo</td>
<td></td>
</tr>
<tr>
<td>Shared::GraphicsEntityShared</td>
<td></td>
</tr>
<tr>
<td>Shared::ModelEntityShared</td>
<td></td>
</tr>
<tr>
<td>StateObjectFeature::StateGraphicsProperty</td>
<td></td>
</tr>
<tr>
<td>StateObjectFeature::StateInfo</td>
<td></td>
</tr>
<tr>
<td>StateObjectFeature::StateObjectFeatureUnit</td>
<td></td>
</tr>
<tr>
<td>StateObjectFeature::StateProperty</td>
<td></td>
</tr>
<tr>
<td>System::AppEntry</td>
<td></td>
</tr>
<tr>
<td>System::ByteOrder</td>
<td></td>
</tr>
<tr>
<td>System::Cpu</td>
<td></td>
</tr>
<tr>
<td>System::SystemInfo</td>
<td></td>
</tr>
<tr>
<td>System::Win32Environment</td>
<td></td>
</tr>
<tr>
<td>System::Win32Registry</td>
<td></td>
</tr>
<tr>
<td>Threading::Barrier</td>
<td></td>
</tr>
<tr>
<td>Threading::CriticalSection</td>
<td></td>
</tr>
<tr>
<td>Threading::Event</td>
<td></td>
</tr>
<tr>
<td>Threading::Interlocked</td>
<td></td>
</tr>
<tr>
<td>Threading::ObjectRef</td>
<td></td>
</tr>
<tr>
<td>Threading::SafeFlag</td>
<td></td>
</tr>
<tr>
<td>Threading::SafePriorityQueue&lt; PRITYPE, TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Threading::SafeQueue&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Threading::Thread</td>
<td></td>
</tr>
<tr>
<td>Threading::ThreadBarrier</td>
<td></td>
</tr>
<tr>
<td>Timing::CalendarTime</td>
<td></td>
</tr>
<tr>
<td>Timing::CalendarBarrier</td>
<td></td>
</tr>
<tr>
<td>Timing::Timer</td>
<td></td>
</tr>
<tr>
<td>Util::Array&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>Util::BitField&lt; NUMBITS &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::Blob</td>
<td></td>
</tr>
<tr>
<td>Util::CommandLineArgs</td>
<td></td>
</tr>
<tr>
<td>Util::Crc</td>
<td></td>
</tr>
<tr>
<td>Util::Delegate&lt; ARGTYPET &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::Dictionary&lt; KEYTYPE, VALUETYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::FixedArray&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::FixedTable&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td></td>
</tr>
<tr>
<td>Util::GlobalStringAtomTable</td>
<td></td>
</tr>
<tr>
<td>Util::GlobalStringAtomTable::DebugInfo</td>
<td>Debug functionality: DebugInfo struct</td>
</tr>
<tr>
<td>Util::Guid</td>
<td></td>
</tr>
<tr>
<td>Util::HashTable&lt; KEYTYPE, VALUETYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::KeyValuePair&lt; KEYTYPE, VALUETYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::List&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::List&lt; TYPE &gt;::Iterator</td>
<td>List iterator</td>
</tr>
<tr>
<td>Util::LocalStringAtomTable</td>
<td></td>
</tr>
<tr>
<td>Util::QuadTree&lt; TYPE &gt;::Node</td>
<td>Node in quad tree</td>
</tr>
<tr>
<td>Util::Queue&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::RandomNumberTable</td>
<td></td>
</tr>
<tr>
<td>Util::RingBuffer&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::RunLengthCodec</td>
<td></td>
</tr>
<tr>
<td>Util::SimpleTree&lt; VALUETYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::SimpleTree&lt; VALUETYPE &gt;::Node</td>
<td>Public node class</td>
</tr>
<tr>
<td>Util::SparseTable&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::Stack&lt; TYPE &gt;</td>
<td></td>
</tr>
<tr>
<td>Util::String</td>
<td></td>
</tr>
<tr>
<td>Util::StringAtom</td>
<td></td>
</tr>
<tr>
<td>Util::StringAtomTableBase</td>
<td></td>
</tr>
<tr>
<td>Util::StringAtomTableBase::StaticString</td>
<td>Static string class for sorting the array</td>
</tr>
<tr>
<td>Util::StringBuffer</td>
<td></td>
</tr>
<tr>
<td>Util::Variant</td>
<td></td>
</tr>
<tr>
<td>Visibility::ObserverContext</td>
<td></td>
</tr>
<tr>
<td>Visibility::VisibilityBoxSystem</td>
<td></td>
</tr>
<tr>
<td>Visibility::VisibilityChecker</td>
<td></td>
</tr>
<tr>
<td>Visibility::VisibilityClusterSystem</td>
<td></td>
</tr>
<tr>
<td>Visibility::VisibilityContainer</td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Visibility::VisibilityContext</td>
<td></td>
</tr>
<tr>
<td>Visibility::VisibilityQuadtree</td>
<td></td>
</tr>
<tr>
<td>Visibility::VisibilityQuery</td>
<td></td>
</tr>
<tr>
<td>Visibility::VisibilitySystemBase</td>
<td></td>
</tr>
<tr>
<td>WeakPtr&lt;TYPE&gt;</td>
<td></td>
</tr>
<tr>
<td>Win32::SysFunc</td>
<td></td>
</tr>
<tr>
<td>Win32::Win32ConsoleHandler</td>
<td></td>
</tr>
<tr>
<td>Win32::Win32Cpu</td>
<td></td>
</tr>
<tr>
<td>Win32::Win32DisplayDevice</td>
<td></td>
</tr>
<tr>
<td>Win32::Win32Guid</td>
<td></td>
</tr>
<tr>
<td>Win32::Win32InputDisplayEventHandler</td>
<td></td>
</tr>
<tr>
<td>Win32::Win32InputServer</td>
<td></td>
</tr>
<tr>
<td>Win32::Win32MiniDump</td>
<td></td>
</tr>
<tr>
<td>Win32::Win32Mouse</td>
<td></td>
</tr>
<tr>
<td>Win32::Win32SkinnedMeshRenderer</td>
<td></td>
</tr>
<tr>
<td>Win32::Win32StringConverter</td>
<td></td>
</tr>
<tr>
<td>Win32::Win32SystemInfo</td>
<td></td>
</tr>
<tr>
<td>Win360::D3D9IndexBuffer</td>
<td></td>
</tr>
<tr>
<td>Win360::D3D9MemoryIndexBufferLoader</td>
<td></td>
</tr>
<tr>
<td>Win360::D3D9MemoryVertexBufferLoader</td>
<td></td>
</tr>
<tr>
<td>Win360::D3D9ShapeRenderer</td>
<td></td>
</tr>
<tr>
<td>Win360::D3D9StreamMeshLoader</td>
<td></td>
</tr>
<tr>
<td>Win360::D3D9StreamTextureSaver</td>
<td></td>
</tr>
<tr>
<td>Win360::D3D9TransformDevice</td>
<td></td>
</tr>
<tr>
<td>Win360::D3D9Types</td>
<td></td>
</tr>
<tr>
<td>Win360::D3D9VertexBuffer</td>
<td></td>
</tr>
<tr>
<td>Win360::D3D9VertexBufferLayout</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360Barrier</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360CalendarTime</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360CriticalSection</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360Event</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360FileTime</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360FSWrapper</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360Heap</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360Interlocked</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360IpAddress</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360MemoryPool</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360Socket</td>
<td></td>
</tr>
<tr>
<td>Win360::Win360Thread</td>
<td></td>
</tr>
</tbody>
</table>
Win360::Win360ThreadBarrier
Win360::Win360ThreadId
Win360::Win360Timer
XInput::XInputGamePad

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:39 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Actions: Action
#include <action.h>

Inheritance diagram for Actions::Action:
Detailed Description

Actions are blocks that request a state change to the entitys, the world. By default the actions are just msgs that have the receiver set and on execute they will just be send to the target entity. The real action execution must be implemented in the property that handles this action msg.

Actions are used by several other DSA subsystems, like the quest and dialog subsystem. Actions could be created from scripting or console.

Actions can be created from a type string by DsaFactoryManager.

(C) 2005 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~<strong>Action ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <strong>OnActivate ()</strong></td>
<td>called when state is activated</td>
</tr>
<tr>
<td>virtual void <strong>OnDeactivate ()</strong></td>
<td>called when state is deactivated</td>
</tr>
<tr>
<td>virtual void <strong>Notify</strong> (const <strong>Ptr</strong>&lt; <strong>Messaging::Message</strong> &gt; &amp;msg)</td>
<td>notify about incoming message</td>
</tr>
<tr>
<td>virtual bool <strong>Start ()</strong></td>
<td>start the action</td>
</tr>
<tr>
<td>virtual void <strong>Stop ()</strong></td>
<td>stop the action</td>
</tr>
<tr>
<td>virtual void <strong>Execute ()</strong></td>
<td>execute the action, by default this sends self as a message to the target entity</td>
</tr>
<tr>
<td>virtual bool <strong>Trigger ()</strong></td>
<td></td>
</tr>
<tr>
<td>virtual void <strong>Assert ()</strong></td>
<td>assert that all required data is present in the world database</td>
</tr>
<tr>
<td>virtual bool <strong>Assert</strong> (const <strong>Ptr</strong>&lt; <strong>Script::InfoLog</strong> &gt; &amp;infoLog)</td>
<td>like <strong>Assert()</strong> but adds errors to the info log object instead of closing the application</td>
</tr>
<tr>
<td>virtual void <strong>ParseArgs</strong> (const <strong>Util::CommandLineArgs</strong> &amp;args)</td>
<td>parse arguments from command line args object</td>
</tr>
<tr>
<td>virtual void <strong>ShowActionInfo ()</strong></td>
<td>make the action show information what would happen if executed; default: do nothing</td>
</tr>
<tr>
<td>virtual Timing::Time <strong>GetTimeLeft ()</strong></td>
<td>get time left (default: 0)</td>
</tr>
<tr>
<td>virtual <strong>Util::String</strong> <strong>GetDebugTxt ()</strong></td>
<td></td>
</tr>
</tbody>
</table>
virtual void SetEntity (const Ptr<Game::Entity> &v)
Set target entity to 'v'.

constPtr<Game::Entity> & GetEntity () const
Target entity if exists.

bool HasEntity () const
Does this contain a target entity?

virtual void Init ()
init after creation, parse args and set entity

virtual void Write (constPtr<Script::ActionReader> &actionReader)
write to action reader

virtual void Read (constPtr<Script::ActionReader> &actionReader)
read from action reader

bool CheckId (const Messaging::Id &id) const
return true if message is of the given id

virtual void Encode (constPtr<IO::BinaryWriter> &writer)
encode message into a stream

virtual void Decode (constPtr<IO::BinaryReader> &reader)
decode message from a stream

void SetHandled (bool b)
set the handled flag

bool Handled () const
return true if the message has been handled

void SetDeferred (bool b)
set deferred flag

bool IsDeferred () const
get deferred flag

void SetDeferredHandled (bool b)
set the deferred handled flag

bool DeferredHandled () const
get the deferred handled flag

int GetRefCount () const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName()</code> const</td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC()</code> const</td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Ptr&lt; Action &gt;</code> <code>CreateActionFromString</code> (const <code>Util::String</code> &amp;cmd)</td>
<td>create complete action from command string</td>
</tr>
<tr>
<td>static <code>Util::Array&lt; Ptr&lt; Action &gt; &gt;</code> <code>CreateActionsFromString</code> (const <code>Util::String</code> &amp;cmd)</td>
<td>create several actions from semicolon-separated commands</td>
</tr>
<tr>
<td>static void <code>DumpRefCountingLeaks</code> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
Ptr< Action >
Actions::Action::CreateActionFromCmdString
(const Util::String cmd ) [static]
```

create complete action from command string

Static method which creates any action object from a command string of the form:

cmd key0=value0 key1=value1 key2=value2

So you would create an action for instance like this:

```cpp
Action::CreateFromCmdString("UnlockQuest quest=Testquest");
```

```cpp
Util::Array< Ptr< Action > >
Actions::Action::CreateActionsFromCmdString
(const Util::String cmd ) [static]
```

create several actions from semicolon-separated commands

Static method which creates many action from a string of the form accepted by `CreateActionFromCmdString()` where several actions are separated by a semicolon.

```cpp
void
Actions::Action::Execute ( ) [virtual]
```

execute the action, by default this sends self as a message to the target entity

As default just send self to target.

Reimplemented in `Actions::ActionList`, `Actions::IfThenElseAction`, and `Actions::SequenceAction`.

```cpp
bool
Actions::Action::Trigger ( ) [virtual]
```
trigger the action, return if the action is still running, by default this calls execute and returns false

Reimplemented in Actions::FSMAction, Actions::IfThenElseAction, and Actions::SequenceAction.

```cpp
void Actions::Action::Assert() [virtual]
```

assert that all required data is present in the world database

Make sure the data required by the action is valid. This may not change the state of the world.

Reimplemented in Actions::ActionList, and Actions::IfThenElseAction.

```cpp
bool Actions::Action::Assert(const Ptr<Script::InfoLog> infoLog) [virtual]
```

like Assert() but adds errors to the info log object instead of closing the application

This method makes sure the data required by the action is valid. This may not change the state of the world. In sub classes errors can be added to the info log object and in case of errors false can be returned instead of closing the application.

Override in subclass! (infoLog in this class is ignored; Assert() will be called; returns always true)

Reimplemented in Actions::IfThenElseAction, and Actions::SequenceAction.

```cpp
void Actions::Action::ParseArgs(const Util::CommandLineArgs &args) [virtual]
```

parse arguments from command line args object

Initialize the action from a CommandLineArgs object. This is necessary for
the automatic scripting support.

```cpp
void Actions::Action::ShowActionInfo() [virtual]
```

make the action show information what would happen if executed; default: do nothing

override in special action

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code
Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Actions::ActionList
Actions::ActionList Class Reference

#include <actionlist.h>

Inheritance diagram for Actions::ActionList:
Detailed Description

contains a list of masterEvents

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void OnActivate ()</td>
<td>called when state is activated</td>
</tr>
<tr>
<td>virtual void OnDeactivate ()</td>
<td>called when state is deactivated</td>
</tr>
<tr>
<td>virtual void Execute ()</td>
<td>execute the actions in action list</td>
</tr>
<tr>
<td>virtual void Assert ()</td>
<td>assert all actions in action list are valid</td>
</tr>
<tr>
<td>virtual bool Assert (Ptr&lt; Script::InfoLog &gt; infoLog)</td>
<td>like Assert() but adds errors to the info log object instead of closing the application</td>
</tr>
<tr>
<td>void SetActionList (const Util::Array&lt; Ptr&lt; Actions::Action &gt; &gt; &amp;a)</td>
<td>set action list</td>
</tr>
<tr>
<td>const Util::Array&lt; Ptr&lt; Actions::Action &gt; &gt; &amp; GetActionList () const</td>
<td>get action list</td>
</tr>
<tr>
<td>virtual void SetEntity (const Ptr&lt; Game::Entity &gt; &amp;v)</td>
<td>Set target entity to <code>v</code>.</td>
</tr>
<tr>
<td>virtual void Write (const Ptr&lt; Script::ActionReader &gt; &amp;actionReader)</td>
<td>write to action reader</td>
</tr>
<tr>
<td>virtual void Read (const Ptr&lt; Script::ActionReader &gt; &amp;actionReader)</td>
<td>read from action reader</td>
</tr>
<tr>
<td>virtual void Init ()</td>
<td>init after creation, parse args and set entity</td>
</tr>
<tr>
<td>virtual void Notify (const Ptr&lt;</td>
<td></td>
</tr>
</tbody>
</table>
### Messaging::Message

- **Start** ()
  - starts the action

- **Stop** ()
  - stops the action

- **Trigger** ()

- **Assert** (const `Ptr<Script::InfoLog>` &infoLog)
  - like **Assert()** but adds errors to the info log object instead of closing the application

- **ParseArgs** (const `Util::CommandLineArgs` &args)
  - parses arguments from command line args object

- **ShowActionInfo** ()
  - makes the action show information what would happen if executed; default: do nothing

- **GetTimeLeft** ()
  - gets time left (default: 0)

- **GetDebugTxt** ()
  - gets current debug txt

- **GetEntity** () const
  - Target entity if exists.

- **HasEntity** () const
  - Does this contain a target entity?

- **CheckId** (const `Messaging::Id` &id) const
  - return true if message is of the given id

- **Encode** (const `Ptr<IO::BinaryWriter>` &writer)
  - encodes message into a stream

- **Decode** (const `Ptr<IO::BinaryReader>` &reader)
  - decodes message from a stream
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetHandled (bool b)</code></td>
<td>set the handled flag</td>
</tr>
<tr>
<td><code>bool Handled () const</code></td>
<td>return true if the message has been handled</td>
</tr>
<tr>
<td><code>void SetDeferred (bool b)</code></td>
<td>set deferred flag</td>
</tr>
<tr>
<td><code>bool IsDeferred () const</code></td>
<td>get deferred flag</td>
</tr>
<tr>
<td><code>void SetDeferredHandled (bool b)</code></td>
<td>set the deferred handled flag</td>
</tr>
<tr>
<td><code>bool DeferredHandled () const</code></td>
<td>get the deferred handled flag</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName</code> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC</code> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static Ptr&lt; Action &gt; CreateActionFromString(const Util::String &amp;cmd)</code></td>
<td>Create complete action from command string</td>
</tr>
<tr>
<td><code>static Util::Array&lt; Ptr&lt; Action &gt; &gt; CreateActionsFromString(const Util::String &amp;cmd)</code></td>
<td>Create several actions from semicolon-separated commands</td>
</tr>
<tr>
<td><code>static void DumpRefCountingLeaks()</code></td>
<td>Dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Actions::ActionList::Execute() [virtual]

execute the actions in action list
execute action in action list
Reimplemented from Actions::Action.
```

```cpp
Ptr< Action >
Actions::Action::CreateActionFromCommandString(const Util::String& cmd) [static, inherited]
```

create complete action from command string

Static method which creates any action object from a command string of the form:

cmd key0=value0 key1=value1 key2=value2

So you would create an action for instance like this:

```cpp
Action::CreateFromCmdString("UnlockQuest quest=Testquest");
```

```cpp
Util::Array< Ptr< Action > >
Actions::Action::CreateActionsFromCommandString(const Util::String& cmd) [static, inherited]
```

create several actions from semicolon-separated commands

Static method which creates many action from a string of the form accepted by `CreateActionFromCommandString()` where several actions are separated by a semicolon.

```cpp
bool Actions::Action::Trigger() [virtual, inherited]
```

trigger the action, return if the action is still running, by default this calls execute and returns false
Reimplemented in Actions::FSMAction, Actions::IfThenElseAction, and Actions::SequenceAction.

```cpp
bool Actions::Action::Assert(const Ptr<Script::InfoLog> infoLog) [virtual, inherited]
```

like Assert() but adds errors to the info log object instead of closing the application.

This method makes sure the data required by the action is valid. This may not change the state of the world. In sub classes errors can be added to the info log object and in case of errors false can be returned instead of closing the application.

Override in subclass! (infoLog in this class is ignored; Assert() will be called; returns always true)

Reimplemented in Actions::IfThenElseAction, and Actions::SequenceAction.

```cpp
void Actions::Action::ParseArgs(const Util::CommandLineArgs& args) [virtual, inherited]
```

parse arguments from command line args object

Initialize the action from a CmdLineArgs object. This is necessary for the automatic scripting support.

```cpp
void Actions::Action::ShowActionInfo() [virtual, inherited]
```

make the action show information what would happen if executed; default: do nothing

override in special action

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount
Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Actions::FSMAction Class Reference

#include <fsmaction.h>

Inheritance diagram for Actions::FSMAction:

```
Core::RefCounted

Messaging::Message

Actions::Action

Actions::FSMAction
```
Detailed Description

A specialized action subclass with support for the behaviour state machine. This just adds an \texttt{OnActivate()} and \texttt{OnDeactivate()} method to allow masterEvents which continue over several frames and need a proper initialization and cleanup point.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <code>Notify</code> (const <code>Ptr&lt; Messaging::Message &gt;</code> &amp;msg)</td>
<td>notify about incoming message</td>
</tr>
<tr>
<td>virtual void <code>OnActivate</code> ()</td>
<td>called when state is activated</td>
</tr>
<tr>
<td>virtual void <code>OnDeactivate</code> ()</td>
<td>called when state is deactivated</td>
</tr>
<tr>
<td>virtual bool <code>Trigger</code> ()</td>
<td>do every frame</td>
</tr>
<tr>
<td>virtual bool <code>Start</code> ()</td>
<td>start the action</td>
</tr>
<tr>
<td>virtual void <code>Stop</code> ()</td>
<td>stop the action</td>
</tr>
<tr>
<td>virtual void <code>Execute</code> ()</td>
<td>execute the action, by default this sends self as a message to the target entity</td>
</tr>
<tr>
<td>virtual void <code>Assert</code> ()</td>
<td>assert that all required data is present in the world database</td>
</tr>
<tr>
<td>virtual bool <code>Assert</code> (const <code>Ptr&lt; Script::InfoLog &gt;</code> &amp;infoLog)</td>
<td>like <code>Assert()</code> but adds errors to the info log object instead of closing the application</td>
</tr>
<tr>
<td>virtual void <code>ParseArgs</code> (const <code>Util::CommandLineArgs</code> &amp;args)</td>
<td>parse arguments from command line args object</td>
</tr>
<tr>
<td>virtual void <code>ShowActionInfo</code> ()</td>
<td>make the action show information what would happen if executed; default: do nothing</td>
</tr>
<tr>
<td>virtual <code>Timing::Time</code> <code>GetTimeLeft</code> ()</td>
<td>get time left (default: 0)</td>
</tr>
<tr>
<td>virtual <code>Util::String</code> <code>GetDebugTxt</code> ()</td>
<td>get current debug txt</td>
</tr>
<tr>
<td>virtual void <code>SetEntity</code> (const <code>Ptr&lt; Game::Entity &gt;</code> &amp;v)</td>
<td>Set target entity to <code>v</code>.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const Ptr&lt; Game::Entity &gt; &amp;</code> GetEntity () const</td>
<td>Target entity if exists.</td>
</tr>
<tr>
<td>bool HasEntity () const</td>
<td>Does this contain a target entity?</td>
</tr>
<tr>
<td>virtual void Init ()</td>
<td>init after creation, parse args and set entity</td>
</tr>
<tr>
<td>virtual void Write (const Ptr&lt; Script::ActionReader &gt; &amp;actionReader)</td>
<td>write to action reader</td>
</tr>
<tr>
<td>virtual void Read (const Ptr&lt; Script::ActionReader &gt; &amp;actionReader)</td>
<td>read from action reader</td>
</tr>
<tr>
<td>bool CheckId (const Messaging::Id &amp;id) const</td>
<td>return true if message is of the given id</td>
</tr>
<tr>
<td>virtual void Encode (const Ptr&lt; IO::BinaryWriter &gt; &amp;writer)</td>
<td>encode message into a stream</td>
</tr>
<tr>
<td>virtual void Decode (const Ptr&lt; IO::BinaryReader &gt; &amp;reader)</td>
<td>decode message from a stream</td>
</tr>
<tr>
<td>void SetHandled (bool b)</td>
<td>set the handled flag</td>
</tr>
<tr>
<td>bool Handled () const</td>
<td>return true if the message has been handled</td>
</tr>
<tr>
<td>void SetDeferred (bool b)</td>
<td>set deferred flag</td>
</tr>
<tr>
<td>bool IsDeferred () const</td>
<td>get deferred flag</td>
</tr>
<tr>
<td>void SetDeferredHandled (bool b)</td>
<td>set the deferred handled flag</td>
</tr>
<tr>
<td>bool DeferredHandled () const</td>
<td>get the deferred handled flag</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td></td>
</tr>
</tbody>
</table>

This documentation describes a set of functions and methods typically found in a C++ class. The functions include methods for initializing and manipulating entities, reading and writing to action readers, encoding and decoding messages, handling and deferring messages, and managing references. The comments provide a brief description of each function's purpose and functionality.
<table>
<thead>
<tr>
<th>void</th>
<th>decrement refcount and destroy object if refcount is zero</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Ptr&lt; Action &gt;</code></td>
<td><code>CreateActionFromString</code> (const <code>Util::String</code> &amp;cmd)</td>
<td>create complete action from command string</td>
</tr>
<tr>
<td>static <code>Util::Array&lt; Ptr&lt; Action &gt; &gt;</code></td>
<td><code>CreateActionsFromString</code> (const <code>Util::String</code> &amp;cmd)</td>
<td>create several actions from semicolon-separated commands</td>
</tr>
<tr>
<td>static void</td>
<td><code>DumpRefCountingLeaks</code> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Actions::FSMAction::Notify (const Ptr<Messaging::Message> msg ) [virtual]
```
notify about incoming message

The state machine will forward any incoming messages to the action.

Reimplemented from `Actions::Action`.

```cpp
Ptr< Action > Actions::Action::CreateActionFromString (const Util::String& cmd ) [static, inherited]
```
create complete action from command string

Static method which creates any action object from a command string of the form:

cmd key0=value0 key1=value1 key2=value2

So you would create an action for instance like this:

```
Action::CreateFromCmdString("UnlockQuest quest=Testquest");
```

```cpp
Util::Array< Ptr< Action > > Actions::Action::CreateActionsFromString (const Util::String& cmd ) [static, inherited]
```
create several actions from semicolon-separated commands

Static method which creates many action from a string of the form accepted by `CreateActionFromString()` where several actions are separated by a semicolon.

```cpp
void Actions::Action::Execute ( ) [virtual, inherited]
```
execute the action, by default this sends self as a message to the
target entity

As default just send self to target.

Reimplemented in Actions::ActionList, Actions::IfThenElseAction, and Actions::SequenceAction.

```cpp
void Actions::Action::Assert() [virtual, inherited]
```

assert that all required data is present in the world database

Make sure the data required by the action is valid. This may not change the state of the world.

Reimplemented in Actions::ActionList, and Actions::IfThenElseAction.

```cpp
bool Actions::Action::Assert(const Ptr<Script::InfoLog> infoLog) [virtual, inherited]
```

like Assert() but adds errors to the info log object instead of closing the application

This method makes sure the data required by the action is valid. This may not change the state of the world. In sub classes errors can be added to the info log object and in case of errors false can be returned instead of closing the application.

Override in subclass! (infoLog in this class is ignored; Assert() will be called; returns always true)

Reimplemented in Actions::IfThenElseAction, and Actions::SequenceAction.

```cpp
void Actions::Action::ParseArgs(const Util::CommandLineArgs & args) [virtual, inherited]
```

parse arguments from command line args object
Initialize the action from a CmdLineArgs object. This is necessary for the automatic scripting support.

```cpp
void Actions::Action::ShowActionInfo() [virtual, inherited]
```

make the action show information what would happen if executed; default: do nothing

override in special action

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

Get the class name of the object.
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Actions::IfThenElseAction
#include <ifthenelseaction.h>

Inheritance diagram for Actions::IfThenElseAction:
Detailed Description

contains a list of masterEvents

(C) 2006 Radon Labs GmbH
### Public Member Functions

- **IfThenElseAction ()**  
  * constructor

- **virtual void Execute ()**  
  * execute the actions in action list

- **virtual bool Trigger ()**  
  * execute the actions in action list

- **virtual void Assert ()**  
  * assert all actions in action list are valid

- **virtual bool Assert (const Ptr<
  Script::InfoLog > &infoLog)**  
  * like Assert() but adds errors to the info log object instead of closing the application

- **void SetCondition (const Ptr<
  Conditions::Condition > &c)**  
  * set condition block

- **const Ptr< Conditions::Condition > & GetCondition () const**  
  * get condition block

- **void SetThenBlock (const Ptr<
  Actions::ActionList > &then)**  
  * set ActionList for then

- **const Ptr< Actions::SequenceAction > & GetThenBlock () const**  
  * get then ActionList

- **void SetElseBlock (const Ptr<
  Actions::ActionList > &elseBlock)**  
  * set else ActionList

- **const Ptr< Actions::SequenceAction > & GetElseBlock () const**  
  * get else ActionList

- **bool HasElseBlock () const**  
  * has else block

- **virtual void SetEntity (const Ptr<
  Game::Entity > &v)**  
  * Set target entity to `v`, also sets entity for subsequent actions.

- **Write (const Ptr<
  Actions::ActionList > &then)**  
  * write action list to info log
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void Script::ActionReader &amp;actionReader</td>
<td>write to action reader</td>
</tr>
<tr>
<td>virtual void Read (const Ptr&lt; Script::ActionReader &gt; &amp;actionReader)</td>
<td>read from action reader</td>
</tr>
<tr>
<td>virtual void OnActivate ()</td>
<td>called when state is activated</td>
</tr>
<tr>
<td>virtual void OnDeactivate ()</td>
<td>called when state is deactivated</td>
</tr>
<tr>
<td>virtual void Notify (const Ptr&lt; Messaging::Message &gt; &amp;msg)</td>
<td>notify about incoming message</td>
</tr>
<tr>
<td>virtual bool Start ()</td>
<td>start the action</td>
</tr>
<tr>
<td>virtual void Stop ()</td>
<td>stop the action</td>
</tr>
<tr>
<td>virtual void ParseArgs (const Util::CommandLineArgs &amp;args)</td>
<td>parse arguments from command line args object</td>
</tr>
<tr>
<td>virtual void ShowActionInfo ()</td>
<td>make the action show information what would happen if executed; default: do nothing</td>
</tr>
<tr>
<td>virtual Timing::Time GetTimeLeft ()</td>
<td>get time left (default: 0)</td>
</tr>
<tr>
<td>virtual Util::String GetDebugTxt ()</td>
<td>get current debug txt</td>
</tr>
<tr>
<td>const Ptr&lt; Game::Entity &gt; &amp; GetEntity () const</td>
<td>Target entity if exists.</td>
</tr>
<tr>
<td>bool HasEntity () const</td>
<td>Does this contain a target entity?</td>
</tr>
<tr>
<td>virtual void Init ()</td>
<td>init after creation, parse args and set entity</td>
</tr>
<tr>
<td>bool CheckId (const Messaging::Id &amp;id) const</td>
<td></td>
</tr>
</tbody>
</table>
return true if message is of the given id

virtual void Encode (const Ptr<IO::BinaryWriter> &writer)
encode message into a stream

virtual void Decode (const Ptr<IO::BinaryReader> &reader)
decode message from a stream

void SetHandled (bool b)
set the handled flag

bool Handled () const
return true if the message has been handled

void SetDeferred (bool b)
set deferred flag

bool IsDeferred () const
get deferred flag

void SetDeferredHandled (bool b)
set the deferred handled flag

bool DeferredHandled () const
get the deferred handled flag

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CreateActionFromString</code> (const <code>Util::String</code> &amp;cmd)</td>
<td>static <code>Ptr&lt; Action &gt;</code>&lt;br&gt;create complete action from command string</td>
</tr>
<tr>
<td><code>CreateActionsFromString</code> (const <code>Util::String</code> &amp;cmd)</td>
<td>static <code>Util::Array&lt; Ptr&lt; Action &gt; &gt;</code>&lt;br&gt;create several actions from semicolon-separated commands</td>
</tr>
<tr>
<td><code>DumpRefCountingLeaks</code> ()</td>
<td>static void&lt;br&gt;dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Actions::IfThenElseAction::Execute () [virtual]
execute the actions in action list
execute action
Reimplemented from Actions::Action.

bool Actions::IfThenElseAction::Trigger () [virtual]
execute the actions in action list
trigger action
Reimplemented from Actions::Action.

Ptr< Action > Actions::Action::CreateActionFromString (const Util::String &cmd) [static, inherited]
create complete action from command string
Static method which creates any action object from a command string of the form:
cmd key0=value0 key1=value1 key2=value2
So you would create an action for instance like this:
Action::CreateFromCmdString("UnlockQuest quest=Testquest");

Util::Array< Ptr< Action > > Actions::Action::CreateActionsFromString (const Util::String &cmd) [static, inherited]
create several actions from semicolon-separated commands
Static method which creates many action from a string of the form accepted by `CreateActionFromString()` where several actions are separated by a semicolon.

```cpp
void Actions::Action::ParseArgs (const Util::CommandLineArgs & args) [virtual, inherited]
```

parse arguments from command line args object

Initialize the action from a CmdLineArgs object. This is necessary for the automatic scripting support.

```cpp
void Actions::Action::ShowActionInfo ( ) [virtual, inherited]
```

make the action show information what would happen if executed; default: do nothing

override in special action

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
get the class name

Get the class name of the object.

get the class FourCC code

Get the class FourCC of the object.

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Actions::SequenceAction
# Actions::SequenceAction Class Reference

```cpp
#include <sequenceaction.h>
```

Inheritance diagram for Actions::SequenceAction:
Detailed Description

Executes a sequence of masterEvents.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SequenceAction()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~SequenceAction()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>Notify</strong> (const Ptr<a href="">Messaging::Message</a> &amp;msg)</td>
<td>Notify about incoming message</td>
</tr>
<tr>
<td><strong>OnActivate()</strong></td>
<td>Called when state is activated</td>
</tr>
<tr>
<td><strong>OnDeactivate()</strong></td>
<td>Called when state is deactivated</td>
</tr>
<tr>
<td><strong>Assert</strong> (const Ptr<a href="">Script::InfoLog</a> &amp;infoLog)</td>
<td>Like Assert() but adds errors to the info log object instead of closing the application</td>
</tr>
<tr>
<td><strong>Start()</strong></td>
<td>Start the action</td>
</tr>
<tr>
<td><strong>Stop()</strong></td>
<td>Stop the action</td>
</tr>
<tr>
<td><strong>Trigger()</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Execute()</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AppendAction</strong> (const Ptr&lt;Action&gt; &amp;action, bool isEffectiveAction=true)</td>
<td></td>
</tr>
<tr>
<td><strong>GetActionList() const</strong></td>
<td>Gets the whole action list</td>
</tr>
<tr>
<td><strong>Write</strong> (const Ptr<a href="">Script::ActionReader</a> &amp;actionReader)</td>
<td>Write to action reader</td>
</tr>
<tr>
<td><strong>Read</strong> (const Ptr<a href="">Script::ActionReader</a> &amp;actionReader)</td>
<td>Read from action reader</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void <code>OnLoad()</code></td>
<td>any setup stuff that needs to be done on load</td>
</tr>
<tr>
<td><code>void FromList(const Ptr&lt;Actions::Action&gt;&amp; actionOrList)</code></td>
<td>insert an action, it will flatten an inserted sequence action</td>
</tr>
<tr>
<td><code>void FromList(const Util::Array&lt;Ptr&lt;Actions::Action&gt;&gt;&amp; actions)</code></td>
<td>append an array of actions, it will flatten sequence actions</td>
</tr>
<tr>
<td>virtual <code>Timing::Time GetTimeLeft()</code></td>
<td>get time left of all actions</td>
</tr>
<tr>
<td>virtual void <code>SetEntity(const Ptr&lt;Game::Entity&gt;&amp; v)</code></td>
<td>overridden set entity method to set entities to sequenced actions</td>
</tr>
<tr>
<td><code>void SetSaveActionsFlag(bool s)</code></td>
<td>set save actions flag (whether or not action list should be saved)</td>
</tr>
<tr>
<td><code>bool GetSaveActionsFlag() const</code></td>
<td>get save actions flag (whether or not action list should be saved)</td>
</tr>
<tr>
<td>virtual void <code>Assert()</code></td>
<td>assert that all required data is present in the world database</td>
</tr>
<tr>
<td><code>virtual void ParseArgs(const Util::CommandLineArgs&amp; args)</code></td>
<td>parse arguments from command line args object</td>
</tr>
<tr>
<td><code>virtual void ShowActionInfo()</code></td>
<td>make the action show information what would happen if executed; default: do nothing</td>
</tr>
<tr>
<td>virtual <code>Util::String GetDebugTxt()</code></td>
<td>get current debug txt</td>
</tr>
<tr>
<td><code>const Ptr&lt;Game::Entity&gt; &amp; GetEntity()</code></td>
<td>Target entity if exists.</td>
</tr>
<tr>
<td><code>bool HasEntity()</code></td>
<td>Does this contain a target entity?</td>
</tr>
<tr>
<td><code>virtual void Init()</code></td>
<td>init after creation, parse args and set entity</td>
</tr>
</tbody>
</table>
bool CheckId (const Messaging::Id &id) const
return true if message is of the given id

virtual void Encode (const Ptr< IO::BinaryWriter > &writer)
encode message into a stream

virtual void Decode (const Ptr< IO::BinaryReader > &reader)
decode message from a stream

void SetHandled (bool b)
set the handled flag

bool Handled () const
return true if the message has been handled

void SetDeferred (bool b)
set deferred flag

bool IsDeferred () const
get deferred flag

void SetDeferredHandled (bool b)
set the deferred handled flag

bool DeferredHandled () const
get the deferred handled flag

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc
<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>Util::String &amp;</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Ptr&lt; Action&gt;</code></td>
<td><strong>CreateActionFromString</strong> (const <code>Util::String</code> &amp;cmd)</td>
<td>create complete action from command string</td>
</tr>
<tr>
<td>static <code>Util::Array&lt; Ptr&lt; Action&gt; &gt;</code></td>
<td><strong>CreateActionsFromString</strong> (const <code>Util::String</code> &amp;cmd)</td>
<td>create several actions from semicolon-separated commands</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application</td>
</tr>
</tbody>
</table>

*NEBULA3_DEBUG builds only!*
Protected Member Functions

bool StartCurrAction ()

start the current action, return false if out of actions or action returned false
Member Function Documentation

void Actions::SequenceAction::Notify(const Messaging::Message msg *)& [virtual]

notify about incoming message

The state machine will forward any incoming messages to the action.
Reimplemented from Actions::Action.

bool Actions::SequenceAction::Start() & [virtual]

start the action
Called from action player property if action will be started
Reimplemented from Actions::Action.

void Actions::SequenceAction::Stop() & [virtual]

stop the action
Called from action player property if action will be stoped
Reimplemented from Actions::Action.

bool Actions::SequenceAction::Trigger() & [virtual]

trigger the actions. use this for sequencially executing the actions within e.g. a FSM
Reimplemented from Actions::Action.

void Actions::SequenceAction::Execute() & [virtual]
trigger the actions. will call execute on all actions. use this for action lists that shall instantly be executed.

Reimplemented from **Actions::Action**.

```cpp
void Actions::SequenceAction::AppendAction(const Ptr<Action> &action, bool isEffectiveAction = true) [inline]
```

append an action to the sequence (NOT allowed if action is running) by default the added action will be noted as the effective action (flag) the effective action will be used to evaluate "ShowActionInfow" (s.b.)

```cpp
void Actions::SequenceAction::FromList(const Ptr<Actions::Action> &actionOrList)
```

insert an action, it will flatten an inserted sequence action

```cpp
bool Actions::SequenceAction::StartCurrAction() [protected]
```

start the current action, return false if out of actions or action returned false

Called from action player property if action will be started

```cpp
Ptr< Action > Actions::Action::CreateActionFromString(const Util::String cmd) [static, inherited]
```

create complete action from command string

Static method which creates any action object from a command string of the form:

cmd key0=value0 key1=value1 key2=value2
So you would create an action for instance like this:

Action::CreateFromCmdString("UnlockQuest quest=Testquest");

create several actions from semicolon-separated commands

Static method which creates many action from a string of the form accepted by CreateActionFromString() where several actions are separated by a semicolon.

assert that all required data is present in the world database

Make sure the data required by the action is valid. This may not change the state of the world.

Reimplemented in Actions::ActionList, and Actions::IfThenElseAction.

parse arguments from command line args object

Initialize the action from a CommandLineArgs object. This is necessary for the automatic scripting support.

make the action show information what would happen if executed; default: do nothing

override in special action
int ( ) const [inline, inherited]
Core::RefCounted::GetRefCount

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
AngularPFeedbackLoop Class Reference

#include <angularpfeedbackloop.h>
Detailed Description

A proportional feedback loop with correct angular interpolation.

(C) 2004 RadonLabs GmbH
Animation::<b>AnimEventHandlerBase</b>
Animation::AnimEventHandlerBase
Class Reference

#include <animeventhandlerbase.h>

Inheritance diagram for Animation::AnimEventHandlerBase:
Detailed Description

This is the baseclass for handling animation events.

Create this in Main application and register it via RegisterAnimEventHandler to the AnimEventServer. This will call the Handler to handle the AnimEvents.

Finally throw away the pointer on the application side and unregister the handler via name (wich equuls the category name)!

Use in subclasses just internal graphics code stuff, cause this handler will work at the render thread side once they are attached!!!

(C) 2009 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AnimEventHandlerBase ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~AnimEventHandlerBase ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual bool <code>HandleEvent (const Animation::AnimEventInfo &amp;event)</code></td>
<td>handle an event</td>
</tr>
<tr>
<td>virtual void <code>OnFrame (Timing::Time time)</code></td>
<td>optional on frame method</td>
</tr>
<tr>
<td>void <code>SetCategoryName (const Util::StringAtom &amp;catName)</code></td>
<td>set the category name</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; <code>GetCategoryName ()</code> const</td>
<td>get the category name</td>
</tr>
<tr>
<td>virtual void <code>Open ()</code></td>
<td>called once on startup</td>
</tr>
<tr>
<td>virtual void <code>Close ()</code></td>
<td>called once before shutdown</td>
</tr>
<tr>
<td>bool <code>IsOpen ()</code> const</td>
<td>return true if open</td>
</tr>
<tr>
<td>virtual bool <code>HandleMessage (const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>handle a message, return true if handled</td>
</tr>
<tr>
<td>virtual void <code>DoWork ()</code></td>
<td>optional &quot;per-frame&quot; DoWork method for continuous handlers</td>
</tr>
<tr>
<td>int <code>GetRefCount ()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::String &amp;className)</code></td>
<td>is an instance of the given class name.</td>
</tr>
<tr>
<td>bool</td>
<td>const</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><code>DumpRefCountingLeaks</code> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</code></td>
</tr>
</tbody>
</table>
Member Function Documentation

void Messaging::Handler::Open () [virtual, inherited]
called once on startup

Open the handler. This method is called once after the handler has been attached to a port and before the first call to HandleMessage().

Reimplemented in Debug::DebugHandler, Http::HttpMessageHandler, IO::IoInterfaceHandler, Animation::AnimEventServer, and Graphics::GraphicsHandler.

void Messaging::Handler::Close () [virtual, inherited]
called once before shutdown

Close the handler. This method is called once before the handler is detached from the port.

Reimplemented in Debug::DebugHandler, Http::HttpMessageHandler, IO::IoInterfaceHandler, Animation::AnimEventServer, and Graphics::GraphicsHandler.

bool Messaging::Handler::HandleMessage (const Ptr< Message > & msg ) [virtual, inherited]

handle a message, return true if handled

Derive this method in a subclass to handle specific messages. The method should return true only if a message has been handled.

Reimplemented in Http::HttpMessageHandler, IO::IoInterfaceHandler, Animation::AnimEventServer, Debug::DebugGraphicsHandler, and Graphics::GraphicsHandler.
void Messaging::Handler::DoWork

optional "per-frame" DoWork method for continuous handlers

This is an optional method for handlers which need to do continuous work (like a render thread message handler). This message will be called after messages have been handled.

Reimplemented in Debug::DebugHandler, Http::HttpMessageHandler, Interface::InterfaceHandlerBase, Debug::DebugGraphicsHandler, and Graphics::GraphicsHandler.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC ( ) const [inline, inherited]
Core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
Animation::\texttt{AnimEventInfo}
Animation::AnimEventInfo Class Reference

#include <animeventinfo.h>
Detailed Description

The *AnimEventInfo* has extra information of the animevent, like the animjob from which it is initiated

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AnimEventInfo</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>GetAnimEvent</strong></td>
<td>const GetAnimEvent</td>
</tr>
<tr>
<td><strong>SetAnimEvent</strong></td>
<td>void SetAnimEvent</td>
</tr>
<tr>
<td><strong>GetAnimJobName</strong></td>
<td>const GetAnimJobName</td>
</tr>
<tr>
<td><strong>SetAnimJobName</strong></td>
<td>void SetAnimJobName</td>
</tr>
<tr>
<td><strong>GetWeight</strong></td>
<td>float GetWeight</td>
</tr>
<tr>
<td><strong>SetWeight</strong></td>
<td>void SetWeight</td>
</tr>
<tr>
<td><strong>SetEntityId</strong></td>
<td>void SetEntityId</td>
</tr>
<tr>
<td><strong>GetEntityId</strong></td>
<td>const GetEntityId</td>
</tr>
</tbody>
</table>
## Friends

<table>
<thead>
<tr>
<th>bool</th>
<th>operator== (const AnimEventInfo &amp;a, const AnimEventInfo &amp;b)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>equality operator (time only)</td>
</tr>
<tr>
<td>bool</td>
<td>operator!= (const AnimEventInfo &amp;a, const AnimEventInfo &amp;b)</td>
</tr>
<tr>
<td></td>
<td>inequality operator (time only)</td>
</tr>
<tr>
<td>bool</td>
<td>operator&lt; (const AnimEventInfo &amp;a, const AnimEventInfo &amp;b)</td>
</tr>
<tr>
<td></td>
<td>less-than operator (time only)</td>
</tr>
<tr>
<td>bool</td>
<td>operator&gt; (const AnimEventInfo &amp;a, const AnimEventInfo &amp;b)</td>
</tr>
<tr>
<td></td>
<td>greather-than operator (time only)</td>
</tr>
<tr>
<td>bool</td>
<td>operator&lt;= (const AnimEventInfo &amp;a, const AnimEventInfo &amp;b)</td>
</tr>
<tr>
<td></td>
<td>less-or-equal operator (time only)</td>
</tr>
<tr>
<td>bool</td>
<td>operator&gt;= (const AnimEventInfo &amp;a, const AnimEventInfo &amp;b)</td>
</tr>
<tr>
<td></td>
<td>greather-or-equal operator (time only)</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Animation::AnimEventServer
Animation::AnimEventServer Class Reference

#include <animeventserver.h>

Inheritance diagram for Animation::AnimEventServer:
Detailed Description

This is the server, which is triggered if a animation event is emitted.

Attach here some handlers to handle special animevents. Handler can be registered via `RegisterAnimEventHandler` and unregistered through `UnregisterAnimEventHandler` messages from the GraphicsProtocol!

Animeventhandler are specified by there category name, they handle!

(C) 2009 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AnimEventServer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~AnimEventServer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Open ()</strong></td>
<td>open the server</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>close the server</td>
</tr>
<tr>
<td><strong>IsOpen () const</strong></td>
<td>return true if open</td>
</tr>
<tr>
<td><strong>OnFrame (Timing::Time time)</strong></td>
<td>delegate to attached handler</td>
</tr>
<tr>
<td><strong>RegisterAnimEventHandler (const Ptr&lt;AnimEventHandlerBase&gt; &amp;newHandler)</strong></td>
<td>attach an animeventhandler</td>
</tr>
<tr>
<td><strong>UnregisterAnimEventHandler (const Util::StringAtom &amp;categoryName)</strong></td>
<td>detach an animeventhandler</td>
</tr>
<tr>
<td><strong>HandleAnimEvents (const Util::Array<a href="">Animation::AnimEventInfo</a> &amp;eventz)</strong></td>
<td>delegate this event to a attached handler</td>
</tr>
<tr>
<td><strong>HandleMessage (const Ptr&lt;Message&gt; &amp;msg)</strong></td>
<td>handle a message, return true if handled</td>
</tr>
<tr>
<td>virtual <strong>DoWork ()</strong></td>
<td>optional &quot;per-frame&quot; DoWork method for continuous handlers</td>
</tr>
<tr>
<td>int <strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>bool IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>bool IsA (const Util::String &amp;rttiName) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>bool IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetClassName () const</strong></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC GetClassFourCC () const</strong></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

`void Messaging::Handler::DoWork ( ) [virtual, inherited]`

optional "per-frame" DoWork method for continuous handlers

This is an optional method for handlers which need to do continuous work (like a render thread message handler). This message will be called after messages have been handled.

Reimplemented in `Debug::DebugHandler`, `Http::HttpMessageHandler`, `Interface::InterfaceHandlerBase`, `Debug::DebugGraphicsHandler`, and `Graphics::GraphicsHandler`.

`int Core::RefCounted::GetRefCount ( ) const [inline, inherited]`

get the current refcount

Return the current refcount of the object.

`void Core::RefCounted::AddRef ( ) [inline, inherited]`

increment refcount by one

Increment the refcount of the object.

`void Core::RefCounted::Release ( ) [inline, inherited]`

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

`const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]`

get the class name
Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Animation::\texttt{AnimJob}
Animation::AnimJob Class Reference

#include <animjob.h>

Inheritance diagram for Animation::AnimJob:

```
Core::RefCounted

Animation::AnimJob

Animation::PlayClipJob
```
Detailed Description

Describes a single animation sampling job in the AnimController. AnimJob objects have a start time and a duration and are arranged in parallel tracks. The sampling results of parallel AnimJobs at a given point in time are mixed into a single resulting animation by the AnimController. Subclasses of AnimJob are used to implement specific tasks like a lookat-controller, IK, and so forth...

FIXME: the current implementation of setting an absolute evaluation time doesn't allow to manipulate the playback speed (for this, advancing the time by a relative amount would be better).

(C) 2008 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnimJob ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~AnimJob ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void SetName (const Util::StringAtom &amp;id)</td>
<td>set human readable name (only used for debugging)</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; GetName () const</td>
<td>get human readable name (only used for debugging)</td>
</tr>
<tr>
<td>bool IsAttachedToSequencer () const</td>
<td>return true if the job is currently attached to a sequencer</td>
</tr>
<tr>
<td>bool IsActive (Timing::Tick time) const</td>
<td>return true if the job has currently playing (EvalTime within start/end time)</td>
</tr>
<tr>
<td>bool IsPending (Timing::Tick time) const</td>
<td>return true if the job has been queued for playback but has not started yet</td>
</tr>
<tr>
<td>bool IsStoppingOrExpired (Timing::Tick time) const</td>
<td>return true if anim job is stopping or expired</td>
</tr>
<tr>
<td>bool IsExpired (Timing::Tick time) const</td>
<td>return true when the job has expired</td>
</tr>
<tr>
<td>void SetTrackIndex (IndexT trackIndex)</td>
<td>set track index, defines blend priority and relationship to other jobs on same track</td>
</tr>
<tr>
<td>IndexT GetTrackIndex () const</td>
<td>get track index</td>
</tr>
<tr>
<td>void SetEnqueueMode (AnimJobEnqueueMode::Code enqueueMode)</td>
<td>set the enqueue behaviour of the new job (default is intercept)</td>
</tr>
<tr>
<td>AnimJobEnqueueMode::Code GetEnqueueMode () const</td>
<td>get the enqueue behaviour of the new job</td>
</tr>
<tr>
<td>void SetExclusiveTag (IndexT id)</td>
<td>exclusive tag (for</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>IndexT</strong></td>
<td><strong>GetExclusiveTag</strong> (const exclusive flag set?)</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetStartTime</strong> (<em>Timing::Tick</em> time) set the start time of the anim job</td>
</tr>
<tr>
<td><strong>Timing::Tick</strong></td>
<td><strong>GetStartTime</strong> (const) get the start time of the anim job (relative to</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetDuration</strong> (<em>Timing::Tick</em> time) set the duration of the anim job (0 ==</td>
</tr>
<tr>
<td><strong>Timing::Tick</strong></td>
<td><strong>GetDuration</strong> (const) get the duration of the anim job</td>
</tr>
<tr>
<td><strong>bool</strong></td>
<td><strong>IsInfinite</strong> (const) return true if the anim job is infinite</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetFadeInTime</strong> (<em>Timing::Tick</em> fadeInTime) set the fade-in time of the</td>
</tr>
<tr>
<td><strong>Timing::Tick</strong></td>
<td><strong>GetFadeInTime</strong> (const) get the fade-in time of the anim job</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetFadeOutTime</strong> (<em>Timing::Tick</em> fadeOutTime) set the fade-out time of</td>
</tr>
<tr>
<td><strong>Timing::Tick</strong></td>
<td><strong>GetFadeOutTime</strong> (const) get the fade-out time of the anim job</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetTimeFactor</strong> (<em>float</em> timeFactor) set time factor</td>
</tr>
<tr>
<td><strong>float</strong></td>
<td><strong>GetTimeFactor</strong> (const) get time factor</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetTimeOffset</strong> (<em>Timing::Tick</em> timeOffset) set sample time offset (if</td>
</tr>
<tr>
<td><strong>Timing::Tick</strong></td>
<td><strong>GetTimeOffset</strong> (const) get sample time offset</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetBlendWeight</strong> (<em>float</em> w) set blend weight of the anim job (default is</td>
</tr>
<tr>
<td><strong>float</strong></td>
<td><strong>GetBlendWeight</strong> (const) get blend weight of the anim job</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetAbsoluteStartTime</strong> () const</td>
<td>get the absolute start time ((\text{BaseTime} + \text{StartTime}))</td>
</tr>
<tr>
<td><strong>GetAbsoluteEndTime</strong> () const</td>
<td>get the absolute end time ((\text{BaseTime} + \text{StartTime} + \text{Duration}))</td>
</tr>
<tr>
<td><strong>GetAbsoluteStopTime</strong> () const</td>
<td>get the absolute, computed end time ((\text{BaseTime} + \text{StartTime} + \text{Duration}) - \text{FadeOutTime})</td>
</tr>
<tr>
<td><strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><strong>GetClassName</strong> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>GetClassFourCC</strong> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetBaseTime(Timing::Tick time)</code></td>
<td>Set the base time of the anim job (set by sequencer when job is attached)</td>
</tr>
<tr>
<td><code>GetBaseTime()</code></td>
<td>Get the base time of the anim job</td>
</tr>
<tr>
<td><code>OnAttachedToSequencer(const AnimSequencer &amp;animSequencer)</code></td>
<td>Called when attached to anim sequencer</td>
</tr>
<tr>
<td><code>OnRemoveFromSequencer()</code></td>
<td>Called when removed from sequencer</td>
</tr>
<tr>
<td><code>ComputeBlendWeight(Timing::Tick relEvalTime)</code></td>
<td>Compute current blend weight, this should take fade-in into account</td>
</tr>
<tr>
<td><code>FixFadeTimes()</code></td>
<td>Fix fade-in/fade-out times if the sum is bigger than the play duration</td>
</tr>
<tr>
<td><code>CreateEvaluationJob(Timing::Tick time, const Ptr&lt;CoreAnimation::AnimSampleBuffer&gt; &amp;mixIn, const Ptr&lt;CoreAnimation::AnimSampleBuffer&gt; &amp;result)</code></td>
<td>Create evaluation job for asynchronous evaluation</td>
</tr>
<tr>
<td><code>EmitAnimEvents(Timing::Tick startTime, Timing::Tick endTime, const Util::String &amp;optionalCategory)</code></td>
<td>Emit anim events inside given time range</td>
</tr>
<tr>
<td><code>UpdateTimes(Timing::Tick time)</code></td>
<td>Compute sample time for next evaluation, always done, also if character isn't visible and no evaluation takes place</td>
</tr>
<tr>
<td><code>Stop(Timing::Tick time)</code></td>
<td>Stop the anim job at the given time</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool Animation::AnimJob::IsActive (Timing::Tick time) const

return true if the job has currently playing (EvalTime within start/end time)

This method will return true if the current eval time is between the start time and end time of the anim job.

bool Animation::AnimJob::IsPending (Timing::Tick time) const

return true if the job has been queued for playback but has not started yet

This method will return true as long as the current eval time is before the start time (the job hasn't started yet).

bool Animation::AnimJob::IsStoppingOrExpired (Timing::Tick time) const

return true if anim job is stopping or expired

Return true if the anim job is currently in or after the fade-out phase.

bool Animation::AnimJob::IsExpired (Timing::Tick time) const

return true when the job has expired

This method will return true if the current eval time is greater then the end time of the job.

Timing::Tick Animation::AnimJob::GetAbsoluteStartTime () const

get the absolute start time (BaseTime + StartTime)
Returns the absolute start time (BaseTime + StartTime).

**Timing::Tick**
Animation::AnimJob::GetAbsoluteEndTime ( ) const

get the absolute end time (BaseTime + StartTime + Duration)

Return the absolute end time (BaseTime + StartTime + Duration). Method fails hard if AnimJob is infinite.

**Timing::Tick**
Animation::AnimJob::GetAbsoluteStopTime ( ) const

get the absolute, computed end time ((BaseTime + StartTime + Duration) - FadeOutTime)

Returns the absolute end time before the fadeout-phase starts ((BaseTime + StartTime + Duration) - FadeOut)

float
Animation::AnimJob::ComputeBlendWeight ( Timing::Tick relEvalTime ) const [protected]

compute current blend weight, this should take fade-in into account

This is a helper method for subclasses and returns the current blend weight for the current relative evaluation time, taking the fade-in and fade-out phases into account.

void
Animation::AnimJob::FixFadeTimes ( ) [protected]

fix fade-in/fade-out times if the sum is bigger then the play duration

This method checks if the fade-in plus the fade-out times would be bigger then the play-duration, if yes it will fix the fade times in order to prevent "blend-popping".

**Ptr< Job >**
Animation::AnimJob::CreateEvaluationJob ( Timing::Tick time, const Ptr< CoreAnimation::AnimSampleBuffer mixIn, > &
create evaluation job for asynchronous evaluation

This method is called by the AnimSequencer when this job is active (the current eval time is between the start and end time of the job). The AnimJob object is expected to fill the provided AnimSampleBuffer with a result (sampled keys and sample counts, the sample counts indicate whether a given sample contributes to the final blended result. If the mixIn pointer is valid, the method must perform animation mixing as well.

This method is usually implemented by subclasses.

```cpp
void Animation::AnimJob::UpdateTimes(Timing::Tick time) [protected, virtual]
```

compute sample time for next evaluation, always done, also if character isn't visible and no evaluation takes place

Updates evaluation times. Must be done every frame, even if character is not visible and so animjob is not evaluated. FIXME: INVISIBLE OBJECTS SHOULD NEVER UPDATE THEIR ANIMATION OR ANIM EVENTS!

```cpp
void Animation::AnimJob::Stop(Timing::Tick time) [protected, virtual]
```

stop the anim job at the given time

Stop the anim job at the given time. This will just update the duration member.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount
Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Animation::AnimJobEnqueueMode
Animation::AnimJobEnqueueMode
Class Reference

#include <animjobenqueue mode.h>
Detailed Description

Describes the behaviour when enqueueing a new animation job into an anim sequencer.

(C) 2010 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enqueue modes</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static const char *</th>
<th><strong>ToString</strong> (Code c)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>convert to string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static Code</th>
<th><strong>FromString</strong> (const char *s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>convert from string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:40 2010
Animation::AnimSequencer
Animation::AnimSequencer Class Reference

#include <animsequencer.h>
Detailed Description

An **AnimSequencer** object arranges AnimJobs along the time line to produce a single, priority-blended result. AnimJobs which are overlapping on the time-line will be blended by the following rules:

- AnimJobs with a higher blend priority dominate lower-priority anim jobs
- If AnimJobs have the same blend priority, the start time of the anim job is used to determine blend priority (jobs which start later dominate jobs which start earlier)

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AnimSequencer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~AnimSequencer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Setup (const Ptr&lt; CoreAnimation::AnimResource &gt; &amp;animResource)</strong></td>
<td>setup the animation controller</td>
</tr>
<tr>
<td><strong>Discard ()</strong></td>
<td>discard the anim sequencer</td>
</tr>
<tr>
<td><strong>IsValid () const</strong></td>
<td>return true if between Setup/Discard</td>
</tr>
<tr>
<td><strong>SetDebugHudEnabled (bool b)</strong></td>
<td>enable/disable the debug hud</td>
</tr>
<tr>
<td><strong>IsDebugHudEnabled () const</strong></td>
<td>get debug hud enabled state</td>
</tr>
<tr>
<td><strong>EnqueueAnimJob (const Ptr&lt; AnimJob &gt; &amp;animJob)</strong></td>
<td>enqueue an anim job</td>
</tr>
<tr>
<td><strong>StopTrack (IndexT trackIndex, bool allowFadeOut=true)</strong></td>
<td>stop all anim jobs on given track</td>
</tr>
<tr>
<td><strong>StopAllTracks (bool allowFadeOut=true)</strong></td>
<td>stop all animations on all tracks</td>
</tr>
<tr>
<td><strong>UpdateTime (Timing::Tick time)</strong></td>
<td>update the animation sequencer time</td>
</tr>
<tr>
<td><strong>StartAsyncEvaluation (const Ptr&lt; Jobs::JobPort &gt; &amp;jobPort)</strong></td>
<td>start asynchronous animation update, returns false if nothing had to be done</td>
</tr>
<tr>
<td><strong>Timing::Tick GetTime () const</strong></td>
<td>get the currently set time</td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreAnimation::AnimSampleBuffer &gt; &amp; GetResult () const</strong></td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Code Snippet</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>get the final sampled result of the last evaluation</td>
<td><code>GetAnimResource</code> () const get pointer to animation resource object</td>
</tr>
<tr>
<td>get pointer to animation resource object</td>
<td><code>GetAnimResource</code> () const get pointer to animation resource object</td>
</tr>
<tr>
<td>get all anim jobs</td>
<td><code>Util::Array&lt; Ptr&lt; AnimJob &gt; &gt;</code> GetAllAnimJobs () const</td>
</tr>
<tr>
<td>get all anim jobs</td>
<td><code>Util::Array&lt; Ptr&lt; AnimJob &gt; &gt;</code> GetAnimJobsByTrackIndex (IndexT trackIndex) const get currently active anim jobs filtered by track index</td>
</tr>
<tr>
<td>get currently active anim jobs filtered by track index</td>
<td><code>Util::Array&lt; Ptr&lt; AnimJob &gt; &gt;</code> GetAnimJobsByTrackIndex (IndexT trackIndex) const get currently active anim jobs filtered by track index</td>
</tr>
<tr>
<td>get anim jobs by name</td>
<td><code>Util::Array&lt; Ptr&lt; AnimJob &gt; &gt;</code> GetAnimJobsByName (const <code>Util::StringAtom</code> &amp;name) const get anim jobs by name</td>
</tr>
<tr>
<td>get anim jobs by name</td>
<td><code>Util::Array&lt; AnimEventInfo &gt;</code> EmitAnimEvents (Timing::Tick startTime, Timing::Tick endTime, bool justDominatingJob, const <code>Util::String</code> &amp;optionalCategory=&quot;&quot;) const FIXME FIXME FIXME: emit anim event infos.</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Animation::AnimSequencer::Discard()

discard the anim sequencer

FIXME: do we have to wait for the anim jobs to finish here??

void Animation::AnimSequencer::EnqueueAnimJob(const Ptr<AnimJob> &animJob)

enqueue an anim job

Enqueue an anim job. This will schedule the anim job for insertion in the next Evaluate(). This deferred handling is necessary because the actual base time of the anim job job is only known in Evaluate() (need to be careful to prevent those pesky one-frame problems).

void Animation::AnimSequencer::StopTrack(IndexT trackIndex, bool allowFadeOut = true)

stop all anim jobs on given track

Stop or cancel all anim jobs on a given track.

void Animation::AnimSequencer::StopAllTracks(bool allowFadeOut = true)

stop all animations on all tracks

Stop or cancel all anim jobs in the sequencer.

void Animation::AnimSequencer::UpdateTime(const Timing::Tick curTime)
update the animation sequencer time

Update the current time of the sequencer. This should be called exactly once per frame, even if the animated object is currently invisible (near the camera but outside the view volume). This will update the anim jobs which have been started or stopped this frame, and it will remove expired anim jobs, but will not sample the animation.

```cpp
bool Animation::AnimSequencer::StartAsyncEvaluation(const Ptr<Jobs::JobPort> &jobPort)
```

start asynchronous animation update, returns false if nothing had to be done

This method should be called once per-frame for each visible animated object AFTER `UpdateTime()` has been called. Actual animation sampling and mixing happens here.

```cpp
Util::Array<AnimEventInfo> Animation::AnimSequencer::EmitAnimEvents(Timing::Tick startTime, Timing::Tick endTime, bool justDominatingJob, const Util::String &optionalCategory ="")
```

FIXMEFIXMEFIXME: emit anim event infos.

Collects all AnimEventInfos of all animjobs which are active in the given time range. If justDominatingJob flag is set, just use the clip with most blend factor.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Animation::<b>PlayClipJob</b>
Animation::PlayClipJob Class Reference

#include <playclipjob.h>

Inheritance diagram for Animation::PlayClipJob:
Detailed Description

An AnimJob which simply plays an animation clip.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayClipJob ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~PlayClipJob ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>void SetClipName (const Util::StringAtom &amp;clipName)</td>
<td>set the anim clip name to play</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; GetClipName () const</td>
<td>get the anim clip name</td>
</tr>
<tr>
<td>void SetLoopCount (float loopCount)</td>
<td>set the loop count (overrides duration)</td>
</tr>
<tr>
<td>float GetLoopCount () const</td>
<td>get the loop count</td>
</tr>
<tr>
<td>void SetName (const Util::StringAtom &amp;id)</td>
<td>set human readable name (only used for debugging)</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; GetName () const</td>
<td>get human readable name (only used for debugging)</td>
</tr>
<tr>
<td>bool IsAttachedToSequencer () const</td>
<td>return true if the job is currently attached to a sequencer</td>
</tr>
<tr>
<td>bool IsActive (Timing::Tick time) const</td>
<td>return true if the job has currently playing (EvalTime within start/end time)</td>
</tr>
<tr>
<td>bool IsPending (Timing::Tick time) const</td>
<td>return true if the job has been queued for playback but has not started yet</td>
</tr>
<tr>
<td>bool IsStoppingOrExpired (Timing::Tick time) const</td>
<td>return true if anim job is stopping or expired</td>
</tr>
<tr>
<td>bool IsExpired (Timing::Tick time) const</td>
<td>return true when the job has expired</td>
</tr>
<tr>
<td>void SetTrackIndex (IndexT trackIndex)</td>
<td>set track index, defines blend priority and relationship to other jobs on same track</td>
</tr>
<tr>
<td>IndexT GetTrackIndex () const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>void SetEnqueueMode (AnimJobEnqueueMode::Code enqueueMode)</td>
<td>set the enqueue behaviour of the new job (default is intercept)</td>
</tr>
<tr>
<td>AnimJobEnqueueMode::Code GetEnqueueMode () const</td>
<td>get the enqueue behaviour of the new job</td>
</tr>
<tr>
<td>void SetExclusiveTag (IndexT id)</td>
<td>exclusive tag (for AnimJobEnqueueMode::IgnoreIfSameExclTagActive)</td>
</tr>
<tr>
<td>IndexT GetExclusiveTag () const</td>
<td>exclusive flag set?</td>
</tr>
<tr>
<td>void SetStartTime (Timing::Tick time)</td>
<td>set the start time of the anim job (relative to base time)</td>
</tr>
<tr>
<td>Timing::Tick GetStartTime () const</td>
<td>get the start time of the anim job (relative to base time)</td>
</tr>
<tr>
<td>void SetDuration (Timing::Tick time)</td>
<td>set the duration of the anim job (0 == infinite)</td>
</tr>
<tr>
<td>Timing::Tick GetDuration () const</td>
<td>get the duration of the anim job</td>
</tr>
<tr>
<td>bool IsInfinite () const</td>
<td>return true if the anim job is infinite</td>
</tr>
<tr>
<td>void SetFadeInTime (Timing::Tick fadeInTime)</td>
<td>set the fade-in time of the anim job</td>
</tr>
<tr>
<td>Timing::Tick GetFadeInTime () const</td>
<td>get the fade-in time of the anim job</td>
</tr>
<tr>
<td>void SetFadeOutTime (Timing::Tick fadeOutTime)</td>
<td>set the fade-out time of the anim job</td>
</tr>
<tr>
<td>Timing::Tick GetFadeOutTime () const</td>
<td>get the fade-out time of the anim job</td>
</tr>
<tr>
<td>void SetTimeFactor (float timeFactor)</td>
<td>set time factor</td>
</tr>
<tr>
<td>float GetTimeFactor () const</td>
<td>get time factor</td>
</tr>
</tbody>
</table>
```plaintext
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetTimeOffset (Timing::Tick timeOffset)</code></td>
<td>set sample time offset (if sampling should not start at the beginning)</td>
</tr>
<tr>
<td><code>Timing::Tick GetTimeOffset () const</code></td>
<td>get sample time offset</td>
</tr>
<tr>
<td><code>void SetBlendWeight (float w)</code></td>
<td>set blend weight of the anim job (default is 1.0)</td>
</tr>
<tr>
<td><code>float GetBlendWeight () const</code></td>
<td>get blend weight of the anim job</td>
</tr>
<tr>
<td><code>Timing::Tick GetAbsoluteStartTime () const</code></td>
<td>get the absolute start time (BaseTime + StartTime)</td>
</tr>
<tr>
<td><code>Timing::Tick GetAbsoluteEndTime () const</code></td>
<td>get the absolute end time (BaseTime + StartTime + Duration)</td>
</tr>
<tr>
<td><code>Timing::Tick GetAbsoluteStopTime () const</code></td>
<td>get the absolute, computed end time (BaseTime + StartTime + Duration) - FadeOutTime</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>SetBaseTime</strong> <em>(Timing::Tick time)</em></td>
<td>set the base time of the anim job (set by sequencer when job is attached)</td>
</tr>
<tr>
<td>Timing::Tick</td>
<td><strong>GetBaseTime</strong> () const</td>
<td>get the base time of the anim job</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnRemoveFromSequencer</strong> ()</td>
<td>called when removed from sequencer</td>
</tr>
<tr>
<td>float</td>
<td><strong>ComputeBlendWeight</strong> <em>(Timing::Tick relEvalTime)</em> const</td>
<td>compute current blend weight, this should take fade-in into account</td>
</tr>
<tr>
<td>void</td>
<td><strong>FixFadeTimes</strong> ()</td>
<td>fix fade-in/fade-out times if the sum is bigger then the play duration</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>UpdateTimes</strong> <em>(Timing::Tick time)</em></td>
<td>compute sample time for next evaluation, always done, also if character isn't visible and no evaluation takes place</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Stop</strong> <em>(Timing::Tick time)</em></td>
<td>stop the anim job at the given time</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool Animation::AnimJob::IsActive (Timing::Tick time ) const [inherited]
return true if the job has currently playing (EvalTime within start/end time)
This method will return true if the current eval time is between the start time and end time of the anim job.

bool Animation::AnimJob::IsPending (Timing::Tick time ) const [inherited]
return true if the job has been queued for playback but has not started yet
This method will return true as long as the current eval time is before the start time (the job hasn't started yet).

bool Animation::AnimJob::IsStoppingOrExpired (Timing::Tick time ) const [inherited]
return true if anim job is stopping or expired
Return true if the anim job is currently in or after the fade-out phase.

bool Animation::AnimJob::IsExpired (Timing::Tick time ) const [inherited]
return true when the job has expired
This method will return true if the current eval time is greater then the end time of the job.

Timing::Tick Animation::AnimJob::GetAbsoluteStartTime ( ) const [inherited]
get the absolute start time (BaseTime + StartTime)
Returns the absolute start time (BaseTime + StartTime).

**Timing::Tick**
Animation::AnimJob::GetAbsoluteEndTime ( ) const [inherited]

get the absolute end time (BaseTime + StartTime + Duration)

Return the absolute end time (BaseTime + StartTime + Duration).
Method fails hard if **AnimJob** is infinite.

**Timing::Tick**
Animation::AnimJob::GetAbsoluteStopTime ( ) const [inherited]

get the absolute, computed end time ((BaseTime + StartTime + Duration) - FadeOutTime)

Returns the absolute end time before the fadeout-phase starts
((BaseTime + StartTime + Duration) - FadeOut)

float
Animation::AnimJob::ComputeBlendWeight ( Timing::Tick relEvalTime ) const [protected, inherited]

compute current blend weight, this should take fade-in into account

This is a helper method for subclasses and returns the current blend weight for the current relative evaluation time, taking the fade-in and fade-out phases into account.

void
Animation::AnimJob::FixFadeTimes ( ) [protected, inherited]

fix fade-in/fade-out times if the sum is bigger then the play duration

This method checks if the fade-in plus the fade-out times would be bigger then the play-duration, if yes it will fix the fade times in order to prevent "blend-popping".

void
Animation::AnimJob::UpdateTimes ( Timing::Tick time ) [protected, virtual, inherited]

compute sample time for next evaluation, always done, also if
character isn't visible and no evaluation takes place

Updates evaluation times. Must be done every frame, even if character is not visible and so animjob is not evaluated. FIXME: INVISIBLE OBJECTS SHOULD NEVER UPDATE THEIR ANIMATION OR ANIM EVENTS!

void Animation::AnimJob::Stop (Timing::Tick time) [protected, virtual, inherited]

stop the anim job at the given time

Stop the anim job at the given time. This will just update the duration member.

int Core::RefCounted::GetRefCount () const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef () [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release () [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName () const [inline, inherited]

get the class name

Get the class name of the object.
Util::FourCC
Core::RefCounted::getClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
• Main Page
• Namespaces
• Data Structures
• Files
• Related Pages

• Alphabetical List
• Data Structures
• Class Hierarchy
• Data Fields

**Animator::AnimatorInstance**
Animator::AnimatorInstance Class Reference

#include <animatornodeinstance.h>
Detailed Description

Legacy N2 crap!

(C) 2008 Radon Labs GmbH
Animator::\texttt{AnimLoopType}
#include <animlooptype.h>
Detailed Description

Legacy N2 stuff!

(C) 2008 RadonLabs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields
AnimatorNode Class Reference

#include <animatornode.h>
Detailed Description

**Legacy** N2 crap!

(C) 2008 RadonLabs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields
AnimKey Class Reference

#include <animkey.h>
Detailed Description

Associate a data value with a point in time.

(C) 2008 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields
AnimKeyArray Class Reference

#include <animkeyarray.h>
Detailed Description

**Legacy** N2 crap!

(C) 2008 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

App::Application
App::Application Class Reference

#include <application.h>

Inheritance diagram for App::Application:
Detailed Description

Provides a simple application model for Nebula3 apps.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Application()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~Application()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>SetCompanyName(const Util::String &amp;n)</code></td>
<td>set company name</td>
</tr>
<tr>
<td><code>GetCompanyName() const</code></td>
<td>get company name</td>
</tr>
<tr>
<td><code>SetAppTitle(const Util::String &amp;n)</code></td>
<td>set application name</td>
</tr>
<tr>
<td><code>GetAppTitle() const</code></td>
<td>get application name</td>
</tr>
<tr>
<td><code>SetAppID(const Util::String &amp;n)</code></td>
<td>set application id</td>
</tr>
<tr>
<td><code>GetAppID() const</code></td>
<td>get application id</td>
</tr>
<tr>
<td><code>SetAppVersion(const Util::String &amp;n)</code></td>
<td>set application version</td>
</tr>
<tr>
<td><code>GetAppVersion() const</code></td>
<td>get application version</td>
</tr>
<tr>
<td><code>SetCmdLineArgs(const Util::CommandLineArgs &amp;a)</code></td>
<td>set command line args</td>
</tr>
<tr>
<td><code>GetCmdLineArgs() const</code></td>
<td>get command line args</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>open the application</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>close the application</td>
</tr>
<tr>
<td><code>Exit()</code></td>
<td>exit the application, call right before leaving main()</td>
</tr>
<tr>
<td><code>Run()</code></td>
<td></td>
</tr>
</tbody>
</table>
run the application, return when user wants to exit

<table>
<thead>
<tr>
<th>bool</th>
<th>IsOpen () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if app is open</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>GetReturnCode () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the return code</td>
</tr>
</tbody>
</table>
Protected Member Functions

```c
void SetReturnCode (int c)
```

set return code
App::ConsoleApplication
#include <consoleapplication.h>

Inheritance diagram for App::ConsoleApplication:
Detailed Description

**Base** class for Nebula3 console applications. Will provide a typical execution environment for console apps.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConsoleApplication ()</strong></td>
</tr>
<tr>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~ConsoleApplication ()</strong></td>
</tr>
<tr>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual bool Open ()</strong></td>
</tr>
<tr>
<td>open the application</td>
</tr>
<tr>
<td><strong>virtual void Close ()</strong></td>
</tr>
<tr>
<td>close the application</td>
</tr>
<tr>
<td><strong>void SetCompanyName (const Util::String &amp;n)</strong></td>
</tr>
<tr>
<td>set company name</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetCompanyName () const</strong></td>
</tr>
<tr>
<td>get company name</td>
</tr>
<tr>
<td><strong>void SetAppTitle (const Util::String &amp;n)</strong></td>
</tr>
<tr>
<td>set application name</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetAppTitle () const</strong></td>
</tr>
<tr>
<td>get application name</td>
</tr>
<tr>
<td><strong>void SetAppID (const Util::String &amp;n)</strong></td>
</tr>
<tr>
<td>set application id</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetAppID () const</strong></td>
</tr>
<tr>
<td>get application id</td>
</tr>
<tr>
<td><strong>void SetAppVersion (const Util::String &amp;n)</strong></td>
</tr>
<tr>
<td>set application version</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetAppVersion () const</strong></td>
</tr>
<tr>
<td>get application version</td>
</tr>
<tr>
<td><strong>void SetCmdLineArgs (const Util::CommandLineArgs &amp;a)</strong></td>
</tr>
<tr>
<td>set command line args</td>
</tr>
<tr>
<td><strong>const Util::CommandLineArgs &amp; GetCmdLineArgs () const</strong></td>
</tr>
<tr>
<td>get command line args</td>
</tr>
<tr>
<td><strong>virtual void Exit ()</strong></td>
</tr>
<tr>
<td>exit the application, call right before leaving main()</td>
</tr>
<tr>
<td><strong>virtual void Run ()</strong></td>
</tr>
<tr>
<td>Function</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>bool IsOpen () const</td>
</tr>
<tr>
<td>int GetReturnCode () const</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetReturnCode</strong> (int c)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set return code</em></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](http://www.doxygen.org) at Fri Mar 26 15:21:40 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

App::GameApplication
App::GameApplication Class Reference

#include <gameapplication.h>

Inheritance diagram for App::GameApplication:

![Inheritance diagram](image)
Detailed Description

Nebula3's default game application. It creates and triggers the GameServer. For game features it creates the core and graphics feature which is used in every game state (such as level game states or only gui game states).

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GameApplication ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~GameApplication ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual bool <strong>Open ()</strong></td>
<td>open the application</td>
</tr>
<tr>
<td>virtual void <strong>Close ()</strong></td>
<td>close the application</td>
</tr>
<tr>
<td>virtual void <strong>Run ()</strong></td>
<td>run the application</td>
</tr>
<tr>
<td>void <strong>AddStateHandler</strong> (const <strong>Ptr</strong>&lt;StateHandler&gt; &amp;state)</td>
<td>add an application state handler</td>
</tr>
<tr>
<td>const <strong>Ptr</strong>&lt; StateHandler &gt; &amp; <strong>FindStateHandlerByName</strong> (const <strong>Util::String</strong> &amp;stateName) const</td>
<td>find a state handler by name</td>
</tr>
<tr>
<td>const <strong>Ptr</strong>&lt; StateHandler &gt; &amp; <strong>GetCurrentStateHandler ()</strong> const</td>
<td>return pointer to current state handler</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetCurrentState ()</strong> const</td>
<td>return state handler of current state</td>
</tr>
<tr>
<td>int <strong>GetNumStates ()</strong> const</td>
<td>get number of application states</td>
</tr>
<tr>
<td>const <strong>Ptr</strong>&lt; StateHandler &gt; &amp; <strong>GetStateHandlerAt</strong> (int index) const</td>
<td>get state handler at index</td>
</tr>
<tr>
<td>void <strong>RequestState</strong> (const <strong>Util::String</strong> &amp;stateName)</td>
<td>request a new state which will be applied at the end of the frame</td>
</tr>
<tr>
<td>void <strong>SetCompanyName</strong> (const <strong>Util::String</strong> &amp;n)</td>
<td>set company name</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetCompanyName ()</strong> const</td>
<td>get company name</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>void SetAppTitle (const Util::String &amp;n)</td>
<td>set application name</td>
</tr>
<tr>
<td>const Util::String &amp; GetAppTitle () const</td>
<td>get application name</td>
</tr>
<tr>
<td>void SetAppID (const Util::String &amp;n)</td>
<td>set application id</td>
</tr>
<tr>
<td>const Util::String &amp; GetAppID () const</td>
<td>get application id</td>
</tr>
<tr>
<td>void SetAppVersion (const Util::String &amp;n)</td>
<td>set application version</td>
</tr>
<tr>
<td>const Util::String &amp; GetAppVersion () const</td>
<td>get application version</td>
</tr>
<tr>
<td>void SetCmdLineArgs (const Util::CommandLineArgs &amp;a)</td>
<td>set command line args</td>
</tr>
<tr>
<td>const Util::CommandLineArgs &amp; GetCmdLineArgs () const</td>
<td>get command line args</td>
</tr>
<tr>
<td>virtual void Exit ()</td>
<td>exit the application, call right before leaving main()</td>
</tr>
<tr>
<td>bool IsOpen () const</td>
<td>return true if app is open</td>
</tr>
<tr>
<td>int GetReturnCode () const</td>
<td>get the return code</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <code>SetupStateHandlers()</code></td>
<td>setup application state handlers</td>
</tr>
<tr>
<td>virtual void <code>CleanupStateHandlers()</code></td>
<td>cleanup application state handlers</td>
</tr>
<tr>
<td>virtual void <code>SetupGameFeatures()</code></td>
<td>setup game features</td>
</tr>
<tr>
<td>virtual void <code>CleanupGameFeatures()</code></td>
<td>cleanup game features</td>
</tr>
<tr>
<td>virtual void <code>DoStateTransition()</code></td>
<td>perform a state transition</td>
</tr>
<tr>
<td>void <code>SetState</code> (const <code>Util::String</code> &amp;s)</td>
<td>set an application state</td>
</tr>
<tr>
<td>virtual void <code>SetupAppFromCmdLineArgs()</code></td>
<td>setup app from cmd lines</td>
</tr>
<tr>
<td>void <code>SetReturnCode</code> (int c)</td>
<td>set return code</td>
</tr>
</tbody>
</table>
void
App::GameApplication::Run ( ) [virtual]

run the application

Run the application. This method will return when the application
wishes to exist.

Reimplemented from App::Application.

void
App::GameApplication::AddStateHandler ( const
                       StateHandler &handler )

add an application state handler

Register a state handler object with the application.

Parameters:
    state  pointer to a state handler object

const Ptr< StateHandler > &
App::GameApplication::FindStateHandlerByName ( Util::String stateName ) const

find a state handler by name

Find a state handler by name.

Parameters:
    name  of state to return the state handler for

Returns:
    pointer to state handler object associated with the state (can be 0)

const Ptr< StateHandler > &
App::GameApplication::GetCurrentStateHandler ( ) const

return pointer to current state handler
Get the current state handler.

```c++
const Util::String &
App::GameApplication::GetCurrentState() const [inline]
```

return state handler of current state

Returns the currently active application state. Can be 0 if no valid state is set.

```c++
void
App::GameApplication::RequestState(const Util::String &stateName)
```

request a new state which will be applied at the end of the frame

Request a new state. This is a public method to switch states (SetState() is private because it invokes some internal voodoo). The requested state will be activated at the end of the frame.

```c++
void
App::GameApplication::SetupStateHandlers()
```

setup application state handlers

Setup the application state handlers. This method is called by App::Open() after the Mangalore subsystems have been initialized. Override this method to create and attach your application state handlers with the application object.

```c++
void
App::GameApplication::CleanupStateHandlers()
```

cleanup application state handlers

Cleanup the application state handlers. This will call the OnRemoveFromApplication() method on all attached state handlers and release them. Usually you don’t need to override this method in your app.

```c++
void
App::GameApplication::SetupGameFeatures()
```


setup game features

Setup new game features which should be used by this application. Overwrite for all features which have to be used.

```cpp
void App::GameApplication::CleanupGameFeatures() [protected, virtual]
```

cleanup game features

Cleanup all added game features

```cpp
void App::GameApplication::DoStateTransition() [protected, virtual]
```

perform a state transition

Do a state transition. This method is called by `SetState()` when the new state is different from the previous state.

```cpp
void App::GameApplication::SetState(const Util::String & s) [protected]
```

set an application state

Set a new application state. This method will call `DoStateTransition()`.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

App::RenderApplication
App::RenderApplication Class Reference

#include <renderapplication.h>

Inheritance diagram for App::RenderApplication:
Detailed Description

Render application class with multithreaded rendering.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RenderApplication ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~RenderApplication ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>void SetLogFileEnabled (bool b)</td>
<td>Enable/disable log file creation</td>
</tr>
<tr>
<td>void SetOverrideRootDirectory (const Util::String &amp;rootDir)</td>
<td>Override root-directory</td>
</tr>
<tr>
<td>void SetMountStandardArchivesEnabled (bool b)</td>
<td>Enable/disable standard archive mounting</td>
</tr>
<tr>
<td>virtual bool Open ()</td>
<td>Open the application</td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td>Close the application</td>
</tr>
<tr>
<td>virtual void Run ()</td>
<td>Run the application</td>
</tr>
<tr>
<td>void SetCompanyName (const Util::String &amp;n)</td>
<td>Set company name</td>
</tr>
<tr>
<td>const Util::String &amp; GetCompanyName () const</td>
<td>Get company name</td>
</tr>
<tr>
<td>void SetAppTitle (const Util::String &amp;n)</td>
<td>Set application name</td>
</tr>
<tr>
<td>const Util::String &amp; GetAppTitle () const</td>
<td>Get application name</td>
</tr>
<tr>
<td>void SetAppID (const Util::String &amp;n)</td>
<td>Set application id</td>
</tr>
<tr>
<td>const Util::String &amp; GetAppID () const</td>
<td>Get application id</td>
</tr>
<tr>
<td>void SetAppVersion (const Util::String &amp;n)</td>
<td>Set application version</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetAppVersion () const</code></td>
<td>get application version</td>
</tr>
<tr>
<td><code>void SetCmdLineArgs (const Util::CommandLineArgs &amp; a)</code></td>
<td>set command line args</td>
</tr>
<tr>
<td><code>const Util::CommandLineArgs &amp; GetCmdLineArgs () const</code></td>
<td>get command line args</td>
</tr>
<tr>
<td><code>virtual void Exit ()</code></td>
<td>exit the application, call right before leaving main()</td>
</tr>
<tr>
<td><code>bool IsOpen () const</code></td>
<td>return true if app is open</td>
</tr>
<tr>
<td><code>int GetReturnCode () const</code></td>
<td>get the return code</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Util::String</th>
<th><strong>LookupProjectDirectory</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lookup the toolkit project key in the registry (<em>Win32 only!</em>)</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetQuitRequested</strong> (bool b)</td>
</tr>
<tr>
<td></td>
<td>set quit requested flag</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsQuitRequested</strong> () const</td>
</tr>
<tr>
<td></td>
<td>return true if quit requested</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnConfigureDisplay</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called to configure display</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnSetupResourceMappers</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called to setup resource mappers</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnProcessInput</strong> ()</td>
</tr>
<tr>
<td></td>
<td>process input (called before rendering)</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnUpdateFrame</strong> ()</td>
</tr>
<tr>
<td></td>
<td>update world</td>
</tr>
<tr>
<td>Timing::Time</td>
<td><strong>GetTime</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get current absolute time</td>
</tr>
<tr>
<td>Timing::Time</td>
<td><strong>GetFrameTime</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get current frame time</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetReturnCode</strong> (int c)</td>
</tr>
<tr>
<td></td>
<td>set return code</td>
</tr>
</tbody>
</table>
Member Function Documentation

void App::RenderApplication::OnSetupResourceMappers( ) [protected, virtual]
called to setup resource mappers

Configure the resource mapper objects for the render thread. NOTE: ResourceMapper objects are created and configured here (the main thread) and then HANDED OVER to the render thread. DO NOT access ResourceMappers after Display::Open() is called. It's best to create ResourceMappers and then immediately forget about them.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**App::ViewerApplication**
App::ViewerApplication Class Reference

#include <viewerapplication.h>

Inheritance diagram for App::ViewerApplication:
Detailed Description

Derived from RenderApplication, adds support for Stages, Views and GraphicsEntities to the asynchronous render app.

(C) 2007 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewerApplication ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~ViewerApplication ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual bool Open ()</td>
<td>open the application</td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td>close the application</td>
</tr>
<tr>
<td>void SetLogFileEnabled (bool b)</td>
<td>enable/disable log file creation</td>
</tr>
<tr>
<td>void SetOverrideRootDirectory (const Util::String &amp;rootDir)</td>
<td>override root-directory</td>
</tr>
<tr>
<td>void SetMountStandardArchivesEnabled (bool b)</td>
<td>enable/disable standard archive mounting</td>
</tr>
<tr>
<td>virtual void Run ()</td>
<td>run the application</td>
</tr>
<tr>
<td>void SetCompanyName (const Util::String &amp;n)</td>
<td>set company name</td>
</tr>
<tr>
<td>const Util::String &amp; GetCompanyName () const</td>
<td>get company name</td>
</tr>
<tr>
<td>void SetAppTitle (const Util::String &amp;n)</td>
<td>set application name</td>
</tr>
<tr>
<td>const Util::String &amp; GetAppTitle () const</td>
<td>get application name</td>
</tr>
<tr>
<td>void SetAppID (const Util::String &amp;n)</td>
<td>set application id</td>
</tr>
<tr>
<td>const Util::String &amp; GetAppID () const</td>
<td>get application id</td>
</tr>
<tr>
<td>void SetAppVersion (const Util::String &amp;n)</td>
<td>set application version</td>
</tr>
<tr>
<td>Function / Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetAppVersion</strong>() const</td>
<td>get application version</td>
</tr>
<tr>
<td>void <strong>SetCmdLineArgs</strong> (const <strong>Util::CommandLineArgs</strong> &amp;a)</td>
<td>set command line args</td>
</tr>
<tr>
<td>const <strong>Util::CommandLineArgs</strong> &amp; <strong>GetCmdLineArgs</strong>() const</td>
<td>get command line args</td>
</tr>
<tr>
<td>virtual void <strong>Exit</strong>()</td>
<td>exit the application, call right before leaving main()</td>
</tr>
<tr>
<td>bool <strong>IsOpen</strong>() const</td>
<td>return true if app is open</td>
</tr>
<tr>
<td>int <strong>GetReturnCode</strong>() const</td>
<td>get the return code</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void</td>
<td>OnProcessInput ()</td>
<td>process input (called before rendering)</td>
</tr>
<tr>
<td>virtual void</td>
<td>OnUpdateFrame ()</td>
<td>update world</td>
</tr>
<tr>
<td>Util::String</td>
<td>LookupProjectDirectory ()</td>
<td>lookup the toolkit project key in the registry <em>(Win32 only!)</em></td>
</tr>
<tr>
<td>void</td>
<td>SetQuitRequested (bool b)</td>
<td>set quit requested flag</td>
</tr>
<tr>
<td>bool</td>
<td>IsQuitRequested () const</td>
<td>return true if quit requested</td>
</tr>
<tr>
<td>virtual void</td>
<td>OnConfigureDisplay ()</td>
<td>called to configure display</td>
</tr>
<tr>
<td>virtual void</td>
<td>OnSetupResourceMappers ()</td>
<td>called to setup resource mappers</td>
</tr>
<tr>
<td>Timing::Time</td>
<td>GetTime () const</td>
<td>get current absolute time</td>
</tr>
<tr>
<td>Timing::Time</td>
<td>GetFrameTime () const</td>
<td>get current frame time</td>
</tr>
<tr>
<td>void</td>
<td>SetReturnCode (int c)</td>
<td>set return code</td>
</tr>
</tbody>
</table>
Member Function Documentation

void App::RenderApplication::OnSetupResourceMappers() [protected, virtual, inherited]
called to setup resource mappers

Configure the resource mapper objects for the render thread. NOTE: ResourceMapper objects are created and configured here (the main thread) and then HANDED OVER to the render thread. DO NOT access ResourceMappers after Display::Open() is called. It's best to create ResourceMappers and then immediately forget about them.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Application::StateHandler
Application::StateHandler Class Reference

#include <statehandler.h>
Detailed Description

State handlers implement actual application state behaviour in subclasses of `Application::StateHandler`. The `Application` class calls state handler objects when a new state is entered, when the current state is left, and for each frame.

State handlers must implement the `OnStateEnter()`, `OnStateLeave()` and `OnStateFrame()` methods accordingly.

(C) 2007 RadonLabs GmbH
Base::ArchiveFileSystemBase
Base::ArchiveFileSystemBase Class Reference

#include <archivefilesystembase.h>
Detailed Description

**Base** class for archive file system wrappers.

(C) 2009 Radon Labs GmbH
Base::CalendarTimeBase
Base::CalendarTimeBase Class Reference

#include <calendartimebase.h>

Inheritance diagram for Base::CalendarTimeBase:
Detailed Description

Allows to obtain the current point in time as year, month, day, etc... down to milliseconds, convert between filetime and CalendarTime, and format the time to a human readable string.

(C) 2007 Radon Labs GmbH
## Public Types

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>Month</td>
</tr>
<tr>
<td></td>
<td>months enum</td>
</tr>
<tr>
<td>enum</td>
<td>Weekday</td>
</tr>
<tr>
<td></td>
<td>weekdays enum</td>
</tr>
<tr>
<td>typedef unsigned int</td>
<td>Year</td>
</tr>
<tr>
<td></td>
<td>typedefs</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CalendarTimeBase()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>SetYear (Year y)</code></td>
<td>Set the year</td>
</tr>
<tr>
<td><code>GetYear () const</code></td>
<td>Get the year</td>
</tr>
<tr>
<td><code>SetMonth (Month m)</code></td>
<td>Set the month</td>
</tr>
<tr>
<td><code>GetMonth () const</code></td>
<td>Get the month</td>
</tr>
<tr>
<td><code>SetWeekday (Weekday wd)</code></td>
<td>Set the day-of-week</td>
</tr>
<tr>
<td><code>GetWeekday () const</code></td>
<td>Get the day-of-week</td>
</tr>
<tr>
<td><code>SetDay (Day d)</code></td>
<td>Set the day (of month)</td>
</tr>
<tr>
<td><code>GetDay () const</code></td>
<td>Get the day (of month)</td>
</tr>
<tr>
<td><code>SetHour (Hour h)</code></td>
<td>Set hour-of-day</td>
</tr>
<tr>
<td><code>GetHour () const</code></td>
<td>Get hour-of-day</td>
</tr>
<tr>
<td><code>SetMinute (Minute m)</code></td>
<td>Set minute-of-hour</td>
</tr>
<tr>
<td><code>GetMinute () const</code></td>
<td>Get minute-of-hour</td>
</tr>
<tr>
<td><code>SetSecond (Second s)</code></td>
<td>Set second-of-minute</td>
</tr>
<tr>
<td><code>GetSecond () const</code></td>
<td>Get second-of-minute</td>
</tr>
<tr>
<td><code>SetMilliSecond (MilliSecond ms)</code></td>
<td>Set milliseconds</td>
</tr>
<tr>
<td><code>GetMilliSecond () const</code></td>
<td>Get milliseconds</td>
</tr>
</tbody>
</table>
get milliseconds
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Timing::CalendarTime</td>
<td><strong>GetSystemTime</strong> ()</td>
<td>get the current system time</td>
</tr>
<tr>
<td>static Timing::CalendarTime</td>
<td><strong>GetLocalTime</strong> ()</td>
<td>get the current local time</td>
</tr>
<tr>
<td>static IO::FileTime</td>
<td><strong>SystemTimeToFileTime</strong> (const Timing::CalendarTime &amp;systemTime)</td>
<td>convert system time to file time</td>
</tr>
<tr>
<td>static Timing::CalendarTime</td>
<td><strong>FileTimeToSystemTime</strong> (const IO::FileTime &amp;fileTime)</td>
<td>convert file time to system time</td>
</tr>
<tr>
<td>static IO::FileTime</td>
<td><strong>LocalTimeToFileTime</strong> (const Timing::CalendarTime &amp;localTime)</td>
<td>convert local time to file time</td>
</tr>
<tr>
<td>static Timing::CalendarTime</td>
<td><strong>FileTimeToLocalTime</strong> (const IO::FileTime &amp;fileTime)</td>
<td>convert file time to local time</td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>Format</strong> (const Util::String &amp;fmtString, const Timing::CalendarTime &amp;calTime)</td>
<td>format to string</td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>MonthToString</strong> (Month m)</td>
<td>convert month to string</td>
</tr>
<tr>
<td>static Month</td>
<td><strong>StringToMonth</strong> (const Util::String &amp;str)</td>
<td>convert string to month</td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>WeekdayToString</strong> (Weekday d)</td>
<td>convert weekday to string</td>
</tr>
<tr>
<td>static Weekday</td>
<td><strong>StringToWeekday</strong> (const Util::String &amp;str)</td>
<td>convert string to weekday</td>
</tr>
</tbody>
</table>
Member Function Documentation

`String
Base::CalendarTimeBase::Format`

```cpp
(const Util::String & fmtString,
 const Timing::CalendarTime calTime)
```

[static]

format to string

Formats a calendar time into a string using the following substitution string:

{YEAR} - the year member
{MONTH} - the month member
{WEEKDAY} - the weekday member
{DAY} - the numerical day-in-month member
{HOUR} - the hour member
{MINUTE} - the minute member
{SECOND} - the second member
{MILLISECOND} - the millisecond member
Base::DisplayDeviceBase
Base::DisplayDeviceBase Class Reference

#include <displaydevicebase.h>

Inheritance diagram for Base::DisplayDeviceBase:
Detailed Description

A DisplayDevice object represents the display where the RenderDevice presents the rendered frame. Use the display device object to get information about available adapters and display modes, and to set the preferred display mode of a Nebula3 application.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DisplayDeviceBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~DisplayDeviceBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>bool AdapterExists (CoreGraphics::Adapter::Code adapter)</strong></td>
<td>Return true if adapter exists</td>
</tr>
<tr>
<td><strong>Util::Array&lt; CoreGraphics::DisplayMode &gt;</strong></td>
<td>GetAvailableDisplayModes (CoreGraphics::Adapter::Code adapter, CoreGraphics::PixelFormat::Code pixelFormat)</td>
</tr>
<tr>
<td><strong>bool SupportsDisplayMode (CoreGraphics::Adapter::Code adapter, const CoreGraphics::DisplayMode &amp;requestedMode)</strong></td>
<td>Return true if a given display mode is supported</td>
</tr>
<tr>
<td><strong>CoreGraphics::DisplayMode</strong></td>
<td>GetCurrentAdapterDisplayMode (CoreGraphics::Adapter::Code adapter)</td>
</tr>
<tr>
<td><strong>CoreGraphics::AdapterInfo</strong></td>
<td>GetAdapterInfo (CoreGraphics::Adapter::Code adapter)</td>
</tr>
<tr>
<td><strong>void SetAdapter (CoreGraphics::Adapter::Code a)</strong></td>
<td>Set display adapter (make sure adapter exists!)</td>
</tr>
<tr>
<td><strong>CoreGraphics::Adapter::Code</strong></td>
<td>GetAdapter () const</td>
</tr>
<tr>
<td><strong>void SetDisplayMode (const CoreGraphics::DisplayMode &amp;m)</strong></td>
<td>Get display adapter</td>
</tr>
<tr>
<td>Function/Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const CoreGraphics::DisplayMode &amp; GetDisplayMode () const</code></td>
<td>Get display mode</td>
</tr>
<tr>
<td><code>void SetAntiAliasQuality (CoreGraphics::AntiAliasQuality::Code aa)</code></td>
<td>Set antialias quality</td>
</tr>
<tr>
<td><code>void SetAntiAliasQuality () const</code></td>
<td>Get antialias quality</td>
</tr>
<tr>
<td><code>void SetAntiAliasQuality (CoreGraphics::AntiAliasQuality::Code aa)</code></td>
<td>Set antialias quality</td>
</tr>
<tr>
<td><code>void SetAntiAliasQuality () const</code></td>
<td>Get antialias quality</td>
</tr>
<tr>
<td><code>bool IsFullscreen () const</code></td>
<td>Get windowed/fullscreen mode</td>
</tr>
<tr>
<td><code>void SetFullscreen (bool b)</code></td>
<td>Set windowed/fullscreen mode</td>
</tr>
<tr>
<td><code>bool IsFullscreen () const</code></td>
<td>Get windowed/fullscreen mode</td>
</tr>
<tr>
<td><code>void SetDisplayModeSwitchEnabled (bool b)</code></td>
<td>Enable display mode switch when running fullscreen (default is true)</td>
</tr>
<tr>
<td><code>bool IsDisplayModeSwitchEnabled () const</code></td>
<td>Is display mode switch enabled for fullscreen?</td>
</tr>
<tr>
<td><code>void SetTripleBufferingEnabled (bool b)</code></td>
<td>Enable triple buffer for fullscreen (default is double buffering)</td>
</tr>
<tr>
<td><code>bool IsTripleBufferingEnabled () const</code></td>
<td>Is triple buffer enabled for fullscreen?</td>
</tr>
<tr>
<td><code>void SetAlwaysOnTop (bool b)</code></td>
<td>Set always-on-top behaviour</td>
</tr>
<tr>
<td><code>bool IsAlwaysOnTop () const</code></td>
<td>Get always-on-top behaviour</td>
</tr>
<tr>
<td><code>void SetVerticalSyncEnabled (bool b)</code></td>
<td>Turn vertical sync on/off</td>
</tr>
<tr>
<td><code>bool IsVerticalSyncEnabled () const</code></td>
<td>Get vertical sync flag</td>
</tr>
<tr>
<td><code>void SetIconName (const Util::String &amp;s)</code></td>
<td>Set optional window icon resource name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetIconName () const</code></td>
<td>Get optional window icon resource name</td>
</tr>
<tr>
<td><code>void SetParentWindow (void *h)</code></td>
<td>Set optional parent window handle</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetParentWindow()</code> const</td>
<td>get optional parent window handle</td>
</tr>
<tr>
<td><code>SetWindowTitle(const Util::String &amp;)</code></td>
<td>set window title string (can be changed anytime)</td>
</tr>
<tr>
<td><code>GetWindowTitle()</code> const</td>
<td>get window title string</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>open the display</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>close the display</td>
</tr>
<tr>
<td><code>IsOpen()</code> const</td>
<td>return true if display is currently open</td>
</tr>
<tr>
<td><code>ProcessWindowMessages()</code></td>
<td>process window system messages, call this method once per frame</td>
</tr>
<tr>
<td><code>AttachEventHandler(const Ptr&lt;CoreGraphics::DisplayEventHandler&gt; &amp;h)</code></td>
<td>attach a display event handler</td>
</tr>
<tr>
<td><code>RemoveEventHandler(const Ptr&lt;CoreGraphics::DisplayEventHandler&gt; &amp;h)</code></td>
<td>remove a display event handler</td>
</tr>
<tr>
<td><code>GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by FourCC</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC const)</code></td>
<td>return true if this object is instance of given class, derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>NotifyEventHandlers</strong> (const CoreGraphics::DisplayEvent &amp;e)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>notify event handlers about an event</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool
Base::DisplayDeviceBase::AdapterExists (CoreGraphics::Adapter::Code adapter)
return true if adapter exists
Checks if the given adapter exists.
Reimplemented in Direct3D9::D3D9DisplayDevice.

Util::Array< DisplayMode >
Base::DisplayDeviceBase::GetAvailableDisplayModes (CoreGraphics::Adapter::Code adapter,
                                                   CoreGraphics::PixelFormat::Code pixel)
get available display modes on given adapter
Returns the display modes on the given adapter in the given pixel format.
Reimplemented in Direct3D9::D3D9DisplayDevice.

bool
Base::DisplayDeviceBase::SupportsDisplayMode (CoreGraphics::Adapter::Code adapter,
                                             const CoreGraphics::DisplayMode & requestedMode)
return true if a given display mode is supported
This method checks the available display modes on the given adapter against the requested display modes and returns true if the display mode exists.
Reimplemented in Direct3D9::D3D9DisplayDevice.

DisplayMode
Base::DisplayDeviceBase::GetCurrentAdapterDisplayMode (CoreGraphics::Adapter::Code adapter)
get current adapter display mode (i.e. the desktop display mode)

This method returns the current adapter display mode. It can be used to get the current desktop display mode.

Reimplemented in Direct3D9::D3D9DisplayDevice.

AdapterInfo
Base::DisplayDeviceBase::GetAdapterInfo ( CoreGraphics::Adapter::Code adapter )

general info about display adapter

Returns information about the provided adapter.

Reimplemented in Direct3D9::D3D9DisplayDevice.

void
Base::DisplayDeviceBase::SetWindowTitle ( const Util::String& str )

window title string (can be changed anytime)

Set the window title. An application should be able to change the window title at any time, that's why this is a virtual method, so that a subclass may override it.

bool
Base::DisplayDeviceBase::Open ( )

open the display

Open the display.

Reimplemented in Win32::Win32DisplayDevice.

void
Base::DisplayDeviceBase::Close ( )

close the display

Close the display.
Reimplemented in `Win32::Win32DisplayDevice`.

```cpp
void
Base::DisplayDeviceBase::ProcessWindowMessages( )
```

process window system messages, call this method once per frame

Process window system messages. Override this method in a subclass.

Reimplemented in `Win32::Win32DisplayDevice`.

```cpp
void
Base::DisplayDeviceBase::AttachEventHandler( const Ptr< CoreGraphics::DisplayEventHandler > & h )
```

attach a display event handler

Attach an event handler to the display device.

```cpp
void
Base::DisplayDeviceBase::RemoveEventHandler( const Ptr< CoreGraphics::DisplayEventHandler > & h )
```

remove a display event handler

Remove an event handler from the display device.

```cpp
bool
Base::DisplayDeviceBase::NotifyEventHandlers( const CoreGraphics::DisplayEvent & e ) [protected]
```

notify event handlers about an event

Notify all event handlers about an event.

```cpp
int
Core::RefCounted::GetRefCount( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Base::GameContentServerBase
Base::GameContentServerBase Class Reference

#include <gamecontentserverbase.h>

Inheritance diagram for Base::GameContentServerBase:
Detailed Description

The game content server initializes access to game content on console platforms. The GameContentServer must be created by the main thread before the first IoServer is created.

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GameContentServerBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~GameContentServerBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void SetTitle (const Util::String &amp;title)</strong></td>
<td>set human readable game title</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetTitle () const</strong></td>
<td>get human readable game title</td>
</tr>
<tr>
<td><strong>void SetTitleId (const Util::String &amp;titleId)</strong></td>
<td>set title id</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetTitleId () const</strong></td>
<td>get title id</td>
</tr>
<tr>
<td><strong>void SetVersion (const Util::String &amp;version)</strong></td>
<td>set title version</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetVersion () const</strong></td>
<td>get title version</td>
</tr>
<tr>
<td><strong>void Setup ()</strong></td>
<td>setup the object</td>
</tr>
<tr>
<td><strong>void Discard ()</strong></td>
<td>discard the object</td>
</tr>
<tr>
<td><strong>bool IsValid () const</strong></td>
<td>return true if object has been setup</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Base::GamePadBase
#include <gamepadbase.h>

Inheritance diagram for Base::GamePadBase:
Detailed Description

An input handler which represents one of at most 4 game pads.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Button</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>gamepad buttons</em></td>
</tr>
<tr>
<td>enum</td>
<td>Axis</td>
</tr>
<tr>
<td></td>
<td><em>gamepad axis</em></td>
</tr>
</tbody>
</table>
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GamePadBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~GamePadBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>bool IsConnected ()</strong> const</td>
<td>return true if this game pad is currently connected</td>
</tr>
<tr>
<td><strong>void SetPlayerIndex (IndexT i)</strong></td>
<td>set player index -&gt; TODO make threadsafe</td>
</tr>
<tr>
<td><strong>IndexT GetPlayerIndex ()</strong> const</td>
<td>get the player index of this game pad</td>
</tr>
<tr>
<td><strong>bool ButtonPressed (Button btn) const</strong></td>
<td>return true if a button is currently pressed</td>
</tr>
<tr>
<td><strong>bool ButtonDown (Button btn) const</strong></td>
<td>return true if button was down at least once in current frame</td>
</tr>
<tr>
<td><strong>bool ButtonUp (Button btn) const</strong></td>
<td>return true if button was up at least once in current frame</td>
</tr>
<tr>
<td><strong>float GetAxisValue (Axis axis) const</strong></td>
<td>get current axis value</td>
</tr>
<tr>
<td><strong>void SetLowFrequencyVibrator (float f)</strong></td>
<td>set low-frequency vibration effect (0.0f .. 1.0f)</td>
</tr>
<tr>
<td><strong>float GetLowFrequencyVibrator () const</strong></td>
<td>get low-frequency vibration</td>
</tr>
<tr>
<td><strong>void SetHighFrequencyVibrator (float f)</strong></td>
<td>set high-frequency vibration effect (0.0f .. 1.0f)</td>
</tr>
<tr>
<td><strong>float GetHighFrequencyVibrator () const</strong></td>
<td>get high-frequency vibration</td>
</tr>
<tr>
<td><strong>Util::Array&lt; Input::InputEvent &gt; GetStateAsInputEvents () const</strong></td>
<td>get current state as an array of input events (override in subclass!)</td>
</tr>
<tr>
<td><strong>bool IsAttached () const</strong></td>
<td>return true if the input handler is currently attached</td>
</tr>
<tr>
<td><strong>virtual void BeginCapture ()</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void EndCapture()</td>
<td>end input capturing to this event handler</td>
</tr>
<tr>
<td>bool IsCapturing() const</td>
<td>return true if this input handler captures input</td>
</tr>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName() const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC() const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Util::String</td>
<td><strong>ButtonAsString</strong> (Button btn)</td>
<td>convert button code to string</td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>AxisAsString</strong> (Axis a)</td>
<td>convert axis to string</td>
</tr>
<tr>
<td>static SizeT</td>
<td><strong>GetMaxNumPlayers</strong> ()</td>
<td>get maximum number of players</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnAttach ()</td>
<td>called when the handler is attached to the input server</td>
</tr>
<tr>
<td>OnReset ()</td>
<td>reset the input handler</td>
</tr>
<tr>
<td>OnRemove ()</td>
<td>called when the handler is removed from the input server</td>
</tr>
<tr>
<td>OnBeginFrame ()</td>
<td>called on InputServer::BeginFrame()</td>
</tr>
<tr>
<td>OnEndFrame ()</td>
<td>called on InputServer::EndFrame();</td>
</tr>
<tr>
<td>OnObtainCapture ()</td>
<td>called when input handler obtains capture</td>
</tr>
<tr>
<td>OnReleaseCapture ()</td>
<td>called when input handler looses capture</td>
</tr>
<tr>
<td>OnEvent (const InputEvent &amp;inputEvent)</td>
<td>called when an input event should be processed</td>
</tr>
</tbody>
</table>
Member Function Documentation

Array<<InputEvent> Base::GamePadBase::GetStateAsInputEvents ( ) const

get current state as an array of input events (override in subclass!)

This method should return the current state of the game pad as input events. It is up to a specific subclass to implement this method.

void Input::InputHandler::BeginCapture ( ) [virtual, inherited]
capture input to this event handler

Begin capturing input to this input handler. This method must be overriden in a subclass, the derived method must call ObtainMouseCapture(), ObtainKeyboardCapture(), or both, depending on what type input events you want to capture. An input handler which captures input gets all input events of the given type exclusively.

Reimplemented in Base::KeyboardBase, and Base::MouseBase.

void Input::InputHandler::EndCapture ( ) [virtual, inherited]
end input capturing to this event handler

End capturing input to this input handler. Override this method in a subclass and release the captures obtained in BeginCapture().

Reimplemented in Base::KeyboardBase, and Base::MouseBase.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Base::IndexBufferBase
Base::IndexBufferBase Class Reference

#include <indexbufferbase.h>

Inheritance diagram for Base::IndexBufferBase:
Detailed Description

A resource which holds an array of indices into an array of vertices.

(C) 2006 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Usage</th>
<th>resource usage flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>State</td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IndexBufferBase ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>virtual ~IndexBufferBase ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void * Map (MapType mapType)</code></td>
<td>map index buffer for CPU access</td>
</tr>
<tr>
<td><code>void Unmap ()</code></td>
<td>unmap the resource</td>
</tr>
<tr>
<td><code>void SetIndexType (CoreGraphics::IndexType::Code type)</code></td>
<td>set the index type (Index16 or Index32)</td>
</tr>
<tr>
<td><code>CoreGraphics::IndexType::Code GetIndexType () const</code></td>
<td>get the index type (Index16 or Index32)</td>
</tr>
<tr>
<td><code>void SetNumIndices (SizeT num)</code></td>
<td>set number of indices</td>
</tr>
<tr>
<td><code>SizeT GetNumIndices () const</code></td>
<td>get number of indices</td>
</tr>
<tr>
<td><code>void SetUsage (Usage usage)</code></td>
<td>set resource usage type</td>
</tr>
<tr>
<td><code>Usage GetUsage () const</code></td>
<td>get resource usage type</td>
</tr>
<tr>
<td><code>void SetAccess (Access access)</code></td>
<td>set resource cpu access type</td>
</tr>
<tr>
<td><code>Access GetAccess () const</code></td>
<td>get cpu access type</td>
</tr>
<tr>
<td><code>void SetAsyncEnabled (bool b)</code></td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td><code>bool IsAsyncEnabled () const</code></td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>void Lock ()</code></td>
<td>set locked to true</td>
</tr>
<tr>
<td><code>void Unlock ()</code></td>
<td></td>
</tr>
</tbody>
</table>
```cpp
set locked to false

bool IsLocked () const
returns true if resource will be used as source for copy process soon

void SetResourceId (const Resourceld &id)
set the resource identifier

cast Resourceld & GetResourceId () const
get the resource identifier

void SetLoader (const Ptr< ResourceLoader > &loader)
set optional resource loader

cast Ptr< ResourceLoader > & GetLoader () const
get optional resource loader

void SetSaver (const Ptr< ResourceSaver > &saver)
set optional resource saver

cast Ptr< ResourceSaver > & GetSaver () const
get optional resource saver

SizeT GetUseCount () const
get current use count

virtual State Load ()
load the resource

virtual void Unload ()
unload the resource, or cancel the pending load

void SetState (State s)
set current state (usually only called during Load()!

State GetState () const
get current state

bool IsLoaded () const
return true if current state is Loaded

bool IsPending () const
return true if current state is Pending

bool LoadFailed () const
return true if current state is Failed

virtual bool Save ()
save the resource

int GetRefCount () const
```
<table>
<thead>
<tr>
<th>get the current refcount</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void</strong> <strong>AddRef</strong> ()</td>
</tr>
<tr>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void</strong> <strong>Release</strong> ()</td>
</tr>
<tr>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool</strong> <strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool</strong> <strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool</strong> <strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>bool</strong> <strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>bool</strong> <strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>bool</strong> <strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><strong>const</strong> <strong>Util::String</strong> &amp; <strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong> <strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>IncrUseCount ()</code></td>
<td>increment use count</td>
</tr>
<tr>
<td>void</td>
<td><code>DecrUseCount ()</code></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void * Base::IndexBufferBase::Map(MapType mapType)
```

map index buffer for CPU access

Make the index buffer content accessible by the CPU. The index buffer must have been initialized with the right Access and Usage flags (see parent class for details). There are several reasons why a mapping the resource may fail, this depends on the platform (for instance, the resource may currently be busy, or selected for rendering).

Reimplemented in `Win360::D3D9IndexBuffer`.

```cpp
void Base::IndexBufferBase::Unmap()
```

unmap the resource

Give up CPU access on the index buffer content.

Reimplemented in `Win360::D3D9IndexBuffer`.

```cpp
Resource::State Resources::Resource::Load() [virtual, inherited]
```

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the `IsPending()` method will return true as long as the load is in progress, and `IsLoaded()` will become true when the loading process has finished. If the load has failed, `IsPending()` will switch to false and `IsLoaded()` will not be true.

```cpp
void Resources::Resource::Unload() [virtual, inherited]
```
unload the resource, or cancel the pending load

This will unload the resource. Only call the method when `IsLoaded()` return true. To cancel a pending asynchronous loading process, call the `CancelPendingLoad()` method.

Reimplemented in CoreAnimation::AnimResource, Base::MeshBase, Base::VertexBufferBase, Direct3D9::D3D9Shader, Direct3D9::D3D9Texture, Win360::D3D9IndexBuffer, Win360::D3D9VertexBuffer, and Models::Model.

```cpp
bool Resources::Resource::Save() [virtual, inherited]
```

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const `Util::String` &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

`Util::FourCC`
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Base::InputServerBase
Base::InputServerBase Class Reference

#include <inputserverbase.h>

Inheritance diagram for Base::InputServerBase:

- Core::RefCounted
  - Base::InputServerBase
    - Win32::Win32InputServer
      - Input::InputServer
Detailed Description

The InputServer is the central object of the Input subsystem. It mainly manages a prioritized list of input handlers which process incoming input events.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InputServerBase</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~InputServerBase</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>SetMaxNumLocalPlayers</strong> (SizeT maxNumLocalPlayers)</td>
<td>set the max number of local players for this application (default is 4)</td>
</tr>
<tr>
<td>SizeT <strong>GetMaxNumLocalPlayers</strong> () const</td>
<td>get the max number of local players</td>
</tr>
<tr>
<td>virtual void <strong>Open</strong> ()</td>
<td>open the input server</td>
</tr>
<tr>
<td>virtual void <strong>Close</strong> ()</td>
<td>close the input server</td>
</tr>
<tr>
<td>bool <strong>IsOpen</strong> () const</td>
<td>return true if open</td>
</tr>
<tr>
<td>void <strong>SetQuitRequested</strong> (bool b)</td>
<td>set the quit requested flag</td>
</tr>
<tr>
<td>bool <strong>IsQuitRequested</strong> () const</td>
<td>return true if some subsystem has requested to quit the app (e.g. Alt-F4)</td>
</tr>
<tr>
<td>void <strong>Reset</strong> ()</td>
<td>reset input state</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; Input::Keyboard &gt;</strong> &amp; <strong>GetDefaultKeyboard</strong> () const</td>
<td>get the default keyboard input handler</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; Input::Mouse &gt;</strong> &amp; <strong>GetDefaultMouse</strong> () const</td>
<td>get the default mouse input handler</td>
</tr>
<tr>
<td><strong>Ptr&lt; Input::GamePad &gt;</strong> <strong>GetDefaultGamePad</strong> (IndexT playerIndex) const</td>
<td>get default gamepad handler (playerIndex is valid up to MaxNumLocalPlayers)</td>
</tr>
<tr>
<td>void <strong>AttachInputHandler</strong> (Input::InputPriority::Code pri, const <strong>Ptr&lt; Input::InputHandler &gt;</strong> &amp; handler)</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| RemoveInputHandler (const Ptr<
  Input::InputHandler > &inputHandler)        | remove an input handler                                                     |
| BeginFrame ()                                 | call before processing window events                                       |
| OnFrame ()                                    | call after processing window events                                        |
| EndFrame ()                                   | call at end of frame                                                        |
| PutEvent (const Input::InputEvent &ie)        | put an input event into the handler chain                                  |
| ClearMouseCapture ()                          | clear the current mouse capture (if exists)                                |
| ClearKeyboardCapture ()                       | clear the current keyboard capture (if exists)                             |
| ClearCapture ()                               | clear both mouse and keyboard captures                                     |
| GetMouseCaptureHandler () const               | return the current mouse capture input handler (return invalid ptr if no capture set) |
| GetKeyboardCaptureHandler () const            | return the current keyboard capture input handler (return invalid ptr if no capture set) |
| ObtainMouseCapture (const Ptr<
  Input::InputHandler > &inputHandler)        | only call from InputHandler: capture mouse input to the given input handler |
| ReleaseMouseCapture (const Ptr<
  Input::InputHandler > &inputHandler)        | only call from InputHandler: release mouse capture                         |
| ObtainKeyboardCapture (const Ptr<
  Input::InputHandler > &inputHandler)        | only call from InputHandler: capture keyboard input to the given input handler |
| ReleaseKeyboardCapture (const Ptr<


<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input::InputHandler</strong> &gt; &amp;inputHandler)</td>
<td>only call from InputHandler: release keyboard capture</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetClassName</strong> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC <strong>GetClassFourCC</strong> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

void Base::InputServerBase::SetMaxNumLocalPlayers(SizeT num)

set the max number of local players for this application (default is 4)

Setup the maximum number of local players for this application. The
default number is 1. This defines the number of game pad objects
created and queried.

void Base::InputServerBase::EndFrame()

call at end of frame

Call this somewhere towards the end of frame, when it is guaraneteed
that noone needs input anymore.

void Base::InputServerBase::PutEvent(const Input::InputEvent & ie)

put an input event into the handler chain

NOTE: MouseMove and RawMouseMove events will be distributed to
all input handlers regardless of mouse capture state!

void Base::InputServerBase::ClearMouseCapture()

clear the current mouse capture (if exists)

This clears the currently set mouse capture (if exists).

void Base::InputServerBase::ClearKeyboardCapture()

clear the current keyboard capture (if exists)

This clears the currently set keyboard capture (if exists).
void Base::InputServerBase::ClearCapture()

clear both mouse and keyboard captures

This clears the mouse and keyboards captures, if set.

void Base::InputServerBase::ObtainMouseCapture(const Ptr<Input::InputHandler> &inputHandler)

only call from InputHandler: capture mouse input to the given input handler

Obtain the mouse capture. All mouse input will go exclusively to the capture input handler until ReleaseMouseCapture() is called.

void Base::InputServerBase::ReleaseMouseCapture(const Ptr<Input::InputHandler> &inputHandler)

only call from InputHandler: release mouse capture

Release the mouse capture. The provided pointer must match the current capture input handler.

void Base::InputServerBase::ObtainKeyboardCapture(const Ptr<Input::InputHandler> &inputHandler)

only call from InputHandler: capture keyboard input to the given input handler

Obtain the keyboard capture. All keyboard input will go exclusively to the capture input handler until ReleaseKeyboardCapture() is called.

void Base::InputServerBase::ReleaseKeyboardCapture(const Ptr<Input::InputHandler> &inputHandler)

only call from InputHandler: release keyboard capture

Release the mouse capture. The provided pointer must match the
current capture input handler.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Base::JobBase
#include <jobbase.h>

Inheritance diagram for Base::JobBase:
Detailed Description

Job objects are asynchronous data crunchers.

A Job is setup from the following components:

- an input data buffer
- an output data buffer
- an uniform data buffer
- the job function

The job function will perform parallel processing of input data into the output buffer. The input data will be split into independent slices and slices will be processed in parallel. The uniform data is identical for every slice and can be used to pass additional arguments to the job function.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~JobBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>void * <strong>AllocPrivateBuffer</strong> (Memory::HeapType heapType, SizeT size)</td>
<td>Allocate a single memory buffer associated with the job, may only be called once, and before <code>Setup()</code>!</td>
</tr>
<tr>
<td>void * <strong>GetPrivateBuffer</strong> () const</td>
<td>Get pointer to job's optional private buffer</td>
</tr>
<tr>
<td>SizeT <strong>GetPrivateBufferSize</strong> () const</td>
<td>Get size of job's optional private buffer</td>
</tr>
<tr>
<td>void <strong>Setup</strong> (const Jobs::JobUniformDesc &amp;uniformDesc, const Jobs::JobDataDesc &amp;inputDesc, const Jobs::JobDataDesc &amp;outputDesc, const Jobs::JobFuncDesc &amp;funcDesc)</td>
<td>Setup the job</td>
</tr>
<tr>
<td>void <strong>Discard</strong> ()</td>
<td>Discard the job</td>
</tr>
<tr>
<td>bool <strong>IsValid</strong> () const</td>
<td>Return true if job has been setup</td>
</tr>
<tr>
<td>void <strong>PatchInputDesc</strong> (const Jobs::JobDataDesc &amp;inputDesc)</td>
<td>Patch input pointers after job has been setup</td>
</tr>
<tr>
<td>void <strong>PatchOutputDesc</strong> (const Jobs::JobDataDesc &amp;outputDesc)</td>
<td>Patch output pointers after job has been setup</td>
</tr>
<tr>
<td>void <strong>PatchUniformDesc</strong> (const Jobs::JobUniformDesc &amp;uniformDesc)</td>
<td>Patch uniform pointer after job has been setup</td>
</tr>
<tr>
<td>const Jobs::JobUniformDesc &amp; <strong>GetUniformDesc</strong> () const</td>
<td>Get uniform data descriptor</td>
</tr>
<tr>
<td>const Jobs::JobDataDesc &amp; <strong>GetInputDesc</strong> () const</td>
<td>Get input data descriptor</td>
</tr>
</tbody>
</table>
get input data descriptor

const Jobs::JobDataDesc & GetOutputDesc () const
get output data descriptor

const Jobs::JobFuncDesc & GetFuncDesc () const
get function descriptor

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Static Public Attributes

```
static const SizeT MaxSliceSize = JobMaxSliceSize

max size of a data slice is 16 kByte - 1 byte
```
### Member Function Documentation

**void * Base::JobBase::AllocPrivateBuffer ( Memory::HeapType heapType, SizeT size )**

allocate a single memory buffer associated with the job, may only be called once, and before **Setup()**!

This method can be used to allocate a single memory buffer, owned by the job object. The method must be called before **Setup()**, and will remain valid until the destructor of the job object is called (so it will survive a **Discard()**).

**int Core::RefCounted::GetRefCount ( ) const [inline, inherited]**

get the current refcount

Return the current refcount of the object.

**void Core::RefCounted::AddRef ( ) [inline, inherited]**

increment refcount by one

Increment the refcount of the object.

**void Core::RefCounted::Release ( ) [inline, inherited]**

decrement refcount and destroy object if refcount is zero

**const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]**

get the class name
Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Base::JobFuncDescBase
Base::JobFuncDescBase Class Reference

#include <jobfuncdescbase.h>

Inheritance diagram for Base::JobFuncDescBase:
Detailed Description

Platform-specific description of a Job function. This can be a simple pointer to a function if the job system works with simple CPU threads, or anything else.

(C) 2009 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Base::JobPortBase
Base::JobPortBase Class Reference

#include <jobportbase.h>

Inheritance diagram for Base::JobPortBase:
Detailed Description

A JobPort accepts Jobs for execution and is used to wait for the completion of jobs or to synchronize the execution of jobs which depend on each other.

(C) 2009 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobPortBase ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~JobPortBase ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void Setup ()</td>
<td>setup the job port</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>discard the job port</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>return true if the job object is valid</td>
</tr>
<tr>
<td>void PushJob (const Ptr&lt; Jobs::Job &gt; &amp;job)</td>
<td>push a job for execution</td>
</tr>
<tr>
<td>void PushJobChain (const Util::Array&lt; Ptr&lt; Jobs::Job &gt; &gt; &amp;jobs)</td>
<td>push a job chain, each job in the chain depends on previous job</td>
</tr>
<tr>
<td>void PushFlush ()</td>
<td>push a flush command (makes sure that jobs don't re-use uniform data from previous jobs)</td>
</tr>
<tr>
<td>void PushSync ()</td>
<td>push a sync command (waits for completion of all previous jobs on this port)</td>
</tr>
<tr>
<td>void WaitDone ()</td>
<td>wait for completion</td>
</tr>
<tr>
<td>bool CheckDone ()</td>
<td>check for completion, return immediately</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>IsInstanceOf (const Util::String &amp;className)</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>const</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount( ) const [inline, inherited]
get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef( ) [inline, inherited]
increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
get the class name

Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Base::KeyboardBase
#include <keyboardbase.h>

Inheritance diagram for Base::KeyboardBase:
Detailed Description

An input handler which represents a keyboard for polling.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KeyboardBase ()</strong></td>
<td>* constructor</td>
</tr>
<tr>
<td>virtual <strong>~KeyboardBase ()</strong></td>
<td>* destructor</td>
</tr>
<tr>
<td>virtual void <strong>BeginCapture ()</strong></td>
<td>* capture input to this event handler</td>
</tr>
<tr>
<td>virtual void <strong>EndCapture ()</strong></td>
<td>* end input capturing to this event handler</td>
</tr>
<tr>
<td>bool <strong>KeyPressed (Input::Key::Code keyCode) const</strong></td>
<td>* return true if a key is currently pressed</td>
</tr>
<tr>
<td>bool <strong>KeyDown (Input::Key::Code keyCode) const</strong></td>
<td>* return true if key was down at least once in current frame</td>
</tr>
<tr>
<td>bool <strong>KeyUp (Input::Key::Code keyCode) const</strong></td>
<td>* return true if key was up at least once in current frame</td>
</tr>
<tr>
<td>const <strong>Util::String &amp; GetCharInput () const</strong></td>
<td>* get character input in current frame</td>
</tr>
<tr>
<td>bool <strong>IsAttached () const</strong></td>
<td>* return true if the input handler is currently attached</td>
</tr>
<tr>
<td>bool <strong>IsCapturing () const</strong></td>
<td>* return true if this input handler captures input</td>
</tr>
<tr>
<td>int <strong>GetRefCount () const</strong></td>
<td>* get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>* increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>* decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>* return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>* return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>* return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
</table>

`dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)`
## Protected Member Functions

<table>
<thead>
<tr>
<th>virtual void</th>
<th><strong>OnAttach</strong> ()</th>
<th>called when the handler is attached to the input server</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void</td>
<td><strong>OnBeginFrame</strong> ()</td>
<td>called on <code>InputServer::BeginFrame()</code></td>
</tr>
<tr>
<td>virtual bool</td>
<td><strong>OnEvent</strong> (const <code>Input::InputEvent</code> &amp;inputEvent)</td>
<td>called when an input event should be processed</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnObtainCapture</strong> ()</td>
<td>called when input handler obtains capture</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnReleaseCapture</strong> ()</td>
<td>called when input handler looses capture</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnReset</strong> ()</td>
<td>reset the input handler</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnRemove</strong> ()</td>
<td>called when the handler is removed from the input server</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnEndFrame</strong> ()</td>
<td>called on <code>InputServer::EndFrame()</code></td>
</tr>
</tbody>
</table>
Member Function Documentation

int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Base::MemoryIndexBufferLoaderBase
Base::MemoryIndexBufferLoaderBase
Class Reference

#include <memoryindexbufferloaderbase.h>

Inheritance diagram for Base::MemoryIndexBufferLoaderBase:
Detailed Description

**Base** resource loader class for initializing an index buffer from data in memory.

(C) 2007 Radon Labs GmbH
Public Member Functions

**MemoryIndexBufferLoaderBase ( )**  
*constructor*

**Setup (CoreGraphics::IndexType::Code**  
indexType, SizeT numIndices, void *indexDataPtr,  
SizeT indexDataSize,  
**CoreGraphics::IndexBuffer::Usage**  
usage=CoreGraphics::IndexBuffer::UsageImmutable,  
**CoreGraphics::IndexBuffer::Access**  
access=CoreGraphics::IndexBuffer::AccessNone)  
*setup index buffer from existing data, or provide 0 pointer if empty*  
*index buffer should be created*

**virtual void**  
**OnAttachToResource (const Ptr< Resource > &res)**  
*called when the resource loader is attached to its resource*

**virtual void**  
**OnRemoveFromResource ( )**  
*called when the resource loader is removed from its resource*

**bool**  
**IsAttachedToResource ( ) const**  
*return true if attached to resource*

**const Ptr< Resource > &**  
**GetResource ( ) const**  
*get pointer to resource*

**virtual bool**  
**CanLoadAsync ( ) const**  
*return true if asynchronous loading is supported*

**virtual bool**  
**OnLoadRequested ( )**  
*called by resource when a load is requested*

**virtual void**  
**OnLoadCancelled ( )**  
*called by resource to cancel a pending load*

**virtual bool**  
**OnPending ( )**  
*call frequently while after OnLoadRequested() to put Resource into loaded state*

**Resource::State**  
**GetState ( ) const**  
*return current state*

**virtual void**  
**Reset ( )**  
*resets loader-stats e.g. state*

**int**  
**GetRefCount ( ) const**
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Protected Member Functions

```cpp
void SetState (Resource::State S)

set current state
```
Member Function Documentation

```cpp
bool Resources::ResourceLoader::CanLoadAsync() const [virtual, inherited]
```

return true if asynchronous loading is supported

This method should be overridden in a subclass and indicates whether the resource loader supports asynchronous resource loading. If asynchronous loading is requested, the `OnLoadRequested()` method will return immediately and the `Resource` object will be put into Pending state. Afterwards, the `Resource` object needs to poll the `ResourceLoader` using the OnPending method, which will eventually setup the `Resource` object.

Reimplemented in `Direct3D9::D3D9StreamShaderLoader`, `Models::StreamModelLoader`, and `Resources::StreamResourceLoader`.

```cpp
bool Resources::ResourceLoader::OnLoadRequested() [virtual, inherited]
```

called by resource when a load is requested

This method is called by our `Resource` object to perform a synchronous or initiate an asynchronous load. When performing a synchronous load, the method should setup the `Resource` and go into the Done state (or Failed state when the load has failed). In asynchronous mode, the method should put the resource loader into the Pending state.

Reimplemented in `CoreGraphics::MemoryMeshLoader`, `Win360::D3D9MemoryIndexBufferLoader`, `Win360::D3D9MemoryVertexBufferLoader`, `Models::StreamModelLoader`, `Resources::D3D9TextureStreamer`, and `Resources::StreamResourceLoader`.

```cpp
void Resources::ResourceLoader::OnLoadCancelled() [virtual, inherited]
```
called by resource to cancel a pending load

This method is called by our Resource object if a pending asynchronous load should be cancelled.

Reimplemented in Models::StreamModelLoader, and Resources::StreamResourceLoader.

```cpp
bool Resources::ResourceLoader::OnPending() [virtual, inherited]
```

call frequently while after OnLoadRequested() to put Resource into loaded state

This method should be called at some time after OnLoadRequested() as long as the ResourceLoader is in the Pending state. This will check whether the asynchronous loader job has finished, and if yes, setup the Resource object, bringing it from the Pending into the Loaded state. If something goes wrong, the ResourceLoader will go into the Failed state. If the outstanding loader job isn't finished yet, the ResourceLoader should remain in Pending state, and the method should return false. Otherwise the Resource should be initialized, and the method should return true.

Reimplemented in Models::StreamModelLoader, and Resources::StreamResourceLoader.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.
void Core::RefCounted::Release( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
get the class name
Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Base::MemoryVertexBufferLoaderBase
Base::MemoryVertexBufferLoaderBase
Class Reference

#include <memoryvertexbufferloaderbase.h>

Inheritance diagram for Base::MemoryVertexBufferLoaderBase:

```
+-----------------+        +-----------------+
| Core::RefCounted |        | Resources::ResourceLoader |
|                  |        | Core::MemoryVertexBufferLoader |
|                  |        +-----------------+
| Core::MemoryVertexBufferLoaderBase |
|                  +-----------------+
| Win32::D3D9MemoryVertexBufferLoader |
+-----------------------------------+
| CoreGraphics::MemoryVertexBufferLoader |
```
Detailed Description

**Base** resource loader class for initializing an vertex buffer from data in memory.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemoryVertexBufferLoaderBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>*<em>Setup (const Util::Array&lt; CoreGraphics::VertexComponent &gt; &amp;vertexComponents, SizeT numVertices, void <em>vertexDataPtr, SizeT vertexDataSize, CoreGraphics::VertexBuffer::Usage usage, CoreGraphics::VertexBuffer::Access access)</em></em></td>
<td>setup vertex buffer data, must remain valid until <code>OnLoadRequested()</code> is called!</td>
</tr>
<tr>
<td><strong>void OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</strong></td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td><strong>void OnRemoveFromResource ()</strong></td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td><strong>bool IsAttachedToResource () const</strong></td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td><strong>const Ptr&lt; Resource &gt; &amp; GetResource () const</strong></td>
<td>get pointer to resource</td>
</tr>
<tr>
<td><strong>virtual bool CanLoadAsync () const</strong></td>
<td>return true if asynchronous loading is supported</td>
</tr>
<tr>
<td><strong>virtual bool OnLoadRequested ()</strong></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td><strong>virtual void OnLoadCancelled ()</strong></td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td><strong>virtual bool OnPending ()</strong></td>
<td>call frequently while after <code>OnLoadRequested()</code> to put <code>Resource</code> into loaded state</td>
</tr>
<tr>
<td><strong>Resource::State GetState () const</strong></td>
<td>return current state</td>
</tr>
<tr>
<td><strong>virtual void Reset ()</strong></td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Protected Member Functions

```c
void SetState (Resource::State S)
```

(set current state)
bool Resources::ResourceLoader::CanLoadAsync() const [virtual, inherited]

return true if asynchronous loading is supported

This method should be overridden in a subclass and indicates whether the resource loader supports asynchronous resource loading. If asynchronous loading is requested, the OnLoadRequested() method will return immediately and the Resource object will be put into Pending state. Afterwards, the Resource object needs to poll the ResourceLoader using the OnPending method, which will eventually setup the Resource object.

Reimplemented in Direct3D9::D3D9StreamShaderLoader, Models::StreamModelLoader, and Resources::StreamResourceLoader.

bool Resources::ResourceLoader::OnLoadRequested() [virtual, inherited]

called by resource when a load is requested

This method is called by our Resource object to perform a synchronous or initiate an asynchronous load. When performing a synchronous load, the method should setup the Resource and go into the Done state (or Failed state when the load has failed). In asynchronous mode, the method should put the resource loader into the Pending state.

Reimplemented in CoreGraphics::MemoryMeshLoader, Win360::D3D9MemoryIndexBufferLoader, Win360::D3D9MemoryVertexBufferLoader, Models::StreamModelLoader, Resources::D3D9TextureStreamer, and Resources::StreamResourceLoader.

void Resources::ResourceLoader::OnLoadCancelled() [virtual, inherited]
called by resource to cancel a pending load

This method is called by our `Resource` object if a pending asynchronous load should be cancelled.

Reimplemented in `Models::StreamModelLoader`, and `Resources::StreamResourceLoader`.

```cpp
bool Resources::ResourceLoader::OnPending() [virtual, inherited]
```

call frequently while after `OnLoadRequested()` to put `Resource` into loaded state

This method should be called at some time after `OnLoadRequested()` as long as the `ResourceLoader` is in the Pending state. This will check whether the asynchronous loader job has finished, and if yes, setup the `Resource` object, bringing it from the Pending into the Loaded state. If something goes wrong, the `ResourceLoader` will go into the Failed state. If the outstanding loader job isn't finished yet, the `ResourceLoader` should remain in Pending state, and the method should return false. Otherwise the `Resource` should be initialized, and the method should return true.

Reimplemented in `Models::StreamModelLoader`, and `Resources::StreamResourceLoader`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.
void Core::RefCounted::Release()
[inline, inherited]

decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName()
const [inline, inherited]

get the class name
Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC()
const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks()
[static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Base::MeshBase
Base::MeshBase Class Reference

#include <meshbase.h>

Inheritance diagram for Base::MeshBase:

```
Core::RefCounted

Resources::Resource

Base::MeshBase

CoreGraphics::Mesh
```
Detailed Description

A mesh maintains a vertex buffer, an optional index buffer and a number of PrimitiveGroup objects. Meshes can be loaded directly from a mesh resource file.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>enum</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MeshBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~MeshBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <strong>Unload ()</strong></td>
<td>unload mesh resource</td>
</tr>
<tr>
<td>bool <strong>HasVertexBuffer ()</strong> const</td>
<td>return true if the mesh has a vertex buffer</td>
</tr>
<tr>
<td>void <strong>SetVertexBuffer (const Ptr&lt; CoreGraphics::VertexBuffer &gt; &amp;vb)</strong></td>
<td>set the vertex buffer object</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; CoreGraphics::VertexBuffer &gt; &amp;</strong></td>
<td><strong>GetVertexBuffer ()</strong> const</td>
</tr>
<tr>
<td>bool <strong>HasIndexBuffer ()</strong> const</td>
<td>return true if the mesh has an index buffer</td>
</tr>
<tr>
<td>void <strong>SetIndexBuffer (const Ptr&lt; CoreGraphics::IndexBuffer &gt; &amp;ib)</strong></td>
<td>set the index buffer object</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; CoreGraphics::IndexBuffer &gt; &amp;</strong></td>
<td><strong>GetIndexBuffer ()</strong> const</td>
</tr>
<tr>
<td>void <strong>SetPrimitiveGroups (const Util::Array&lt; CoreGraphics::PrimitiveGroup &gt; &amp;groups)</strong></td>
<td>set primitive groups</td>
</tr>
<tr>
<td><strong>SizeT GetNumPrimitiveGroups ()</strong> const</td>
<td>get the number of primitive groups in the mesh</td>
</tr>
<tr>
<td>const <strong>CoreGraphics::PrimitiveGroup &amp;</strong></td>
<td><strong>GetPrimitiveGroupAtIndex (IndexT i)</strong> const</td>
</tr>
</tbody>
</table>


void ApplyPrimitives (IndexT primGroupIndex)
apply mesh data for rendering in renderdevice

void SetAsyncEnabled (bool b)
request synchronous/asynchronous resource loading

bool IsAsyncEnabled () const
return true if asynchronous resource loading is enabled

void Lock ()
set locked to true

void Unlock ()
set locked to false

bool IsLocked () const
returns true if resource will be used as source for copy process soon

void SetResourceId (const ResourceId &id)
set the resource identifier

const ResourceId & GetResourceId () const
get the resource identifier

void SetLoader (const Ptr<ResourceLoader> &loader)
set optional resource loader

const Ptr<ResourceLoader> & GetLoader () const
get optional resource loader

void SetSaver (const Ptr<ResourceSaver> &saver)
set optional resource saver

const Ptr<ResourceSaver> & GetSaver () const
get optional resource saver

SizeT GetUseCount () const
get current use count

virtual State Load ()
load the resource

void SetState (State s)
set current state (usually only called during
Load()

State

GetState () const
get current state

bool IsLoaded () const
return true if current state is Loaded

bool IsPending () const
return true if current state is Pending

bool LoadFailed () const
return true if current state is Failed

virtual bool Save ()
save the resource

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

IsA (const Util::FourCC
<table>
<thead>
<tr>
<th>bool</th>
<th>&amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

```
static void DumpRefCountingLeaks ()
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
```
## Protected Member Functions

<table>
<thead>
<tr>
<th></th>
<th>function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>IncrUseCount</strong> ()</td>
<td>increment use count</td>
</tr>
<tr>
<td>void</td>
<td><strong>DecrUseCount</strong> ()</td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Resource::State**  
Resources::Resource::Load() [virtual, inherited]  

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the **IsPending()** method will return true as long as the load is in progress, and **IsLoaded()** will become true when the loading process has finished. If the load has failed, **IsPending()** will switch to false and **IsLoaded()** will not be true.

```cpp
bool Resources::Resource::Save() [virtual, inherited]
```

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

    const Util::String &
    Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

    Util::FourCC
    Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

    void
    Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**Base::MouseBase**
Base::MouseBase Class Reference

#include <mousebase.h>

Inheritance diagram for Base::MouseBase:
Detailed Description

An input handler which represents a mouse for polling.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MouseBase ()</strong></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td><strong>virtual ~MouseBase ()</strong></td>
<td><em>destructor</em></td>
</tr>
<tr>
<td><strong>virtual void BeginCapture ()</strong></td>
<td><em>capture input to this event handler</em></td>
</tr>
<tr>
<td><strong>virtual void EndCapture ()</strong></td>
<td><em>end input capturing to this event handler</em></td>
</tr>
<tr>
<td><strong>bool ButtonPressed (Input::MouseButton::Code btn) const</strong></td>
<td><em>return true if button is currently pressed</em></td>
</tr>
<tr>
<td><strong>bool ButtonDown (Input::MouseButton::Code btn) const</strong></td>
<td><em>return true if button was down at least once in current frame</em></td>
</tr>
<tr>
<td><strong>bool ButtonUp (Input::MouseButton::Code btn) const</strong></td>
<td><em>return true if button was up at least once in current frame</em></td>
</tr>
<tr>
<td><strong>bool ButtonDoubleClicked (Input::MouseButton::Code btn) const</strong></td>
<td><em>return true if a button has been double clicked</em></td>
</tr>
<tr>
<td><strong>bool WheelForward () const</strong></td>
<td><em>return true if mouse wheel rotated forward</em></td>
</tr>
<tr>
<td><strong>bool WheelBackward () const</strong></td>
<td><em>return true if mouse wheel rotated backward</em></td>
</tr>
<tr>
<td><strong>const Math::float2 &amp; GetPixelPosition () const</strong></td>
<td><em>get current absolute mouse position (in pixels)</em></td>
</tr>
<tr>
<td><strong>const Math::float2 &amp; GetScreenPosition () const</strong></td>
<td><em>get current screen space mouse position (0.0 .. 1.0)</em></td>
</tr>
<tr>
<td><strong>const Math::float2 &amp; GetMovement () const</strong></td>
<td><em>get mouse movement</em></td>
</tr>
<tr>
<td><strong>bool IsAttached () const</strong></td>
<td><em>return true if the input handler is currently attached</em></td>
</tr>
<tr>
<td><strong>bool IsCapturing () const</strong></td>
<td><em>return true if this input handler captures input</em></td>
</tr>
<tr>
<td>int</td>
<td><code>GetRefCount()</code> const</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------</td>
</tr>
<tr>
<td>void</td>
<td><code>AddRef()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>Release()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsInstanceOf(const Util::String &amp;className)</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA(const Rtti &amp;rtti)</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA(const Util::String &amp;rttiName)</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code> const</td>
</tr>
<tr>
<td>const Utility::String &amp;</td>
<td><code>GetClassName()</code> const</td>
</tr>
<tr>
<td>Utility::FourCC</td>
<td><code>GetClassFourCC()</code> const</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
# Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>OnAttach</strong> ()</td>
<td>called when the handler is attached to the input server</td>
</tr>
<tr>
<td>virtual void <strong>OnBeginFrame</strong> ()</td>
<td>called on InputServer::BeginFrame()</td>
</tr>
<tr>
<td>virtual bool <strong>OnEvent</strong> (const Input::InputEvent &amp;inputEvent)</td>
<td>called when an input event should be processed</td>
</tr>
<tr>
<td>virtual void <strong>OnObtainCapture</strong> ()</td>
<td>called when input handler obtains capture</td>
</tr>
<tr>
<td>virtual void <strong>OnReleaseCapture</strong> ()</td>
<td>called when input handler looses capture</td>
</tr>
<tr>
<td>virtual void <strong>OnReset</strong> ()</td>
<td>reset the input handler</td>
</tr>
<tr>
<td>virtual void <strong>OnRemove</strong> ()</td>
<td>called when the handler is removed from the input server</td>
</tr>
<tr>
<td>virtual void <strong>OnEndFrame</strong> ()</td>
<td>called on InputServer::EndFrame();</td>
</tr>
</tbody>
</table>
Member Function Documentation

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
get the current refcount
Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
get the class name
Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Base::MouseRenderDeviceBase
Base::MouseRenderDeviceBase Class Reference

#include <mouserenderdevicebase.h>

Inheritance diagram for Base::MouseRenderDeviceBase:
Detailed Description

Renders one (or more, depending on platform) mouse cursors.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MouseRenderDeviceBase()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~MouseRenderDeviceBase()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <code>Setup()</code></td>
<td>Open the mouse renderer</td>
</tr>
<tr>
<td>void <code>Discard()</code></td>
<td>Close the mouse renderer</td>
</tr>
<tr>
<td>bool <code>IsValid()</code></td>
<td>Return true if the mouse renderer is valid</td>
</tr>
<tr>
<td>void <code>PreloadTextures</code></td>
<td>Load mouse pointer textures</td>
</tr>
<tr>
<td>void <code>UpdatePointers</code></td>
<td>Update the mouse renderer</td>
</tr>
<tr>
<td>void <code>RenderPointers()</code></td>
<td>Render mouse pointer(s)</td>
</tr>
<tr>
<td>int <code>GetRefCount()</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf(Rtti)</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf(const Util::String &amp;className)</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA(Rtti)</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
</tbody>
</table>

- **MouseRenderDeviceBase**: Base class for mouse render device.
- **PreloadTextures**: Preloads textures for mouse pointers.
- **UpdatePointers**: Updates mouse pointer data.
- **RenderPointers**: Renders mouse pointers.
- **IsValid**: Checks if the mouse renderer is valid.
- **GetRefCount**: Gets the current refcount.
- **AddRef**: Increments the refcount by one.
- **Release**: Decrements the refcount and destroys the object if the refcount is zero.
- **IsInstanceOf**: Checks if the object is an instance of a given class.
- **IsA**: Checks if the object is an instance of a given class.
<table>
<thead>
<tr>
<th>return true if this object is instance of given class, or a derived class</th>
</tr>
</thead>
</table>
| bool **IsA** (const Util::String &rttiName) const  
_ return true if this object is instance of given class, or a derived class, by string |
| bool **IsA** (const Util::FourCC &rttiFourCC) const  
_ return true if this object is instance of given class, or a derived class, by fourcc |
| const Util::String & **GetClassName** () const  
_ get the class name |
| Util::FourCC **GetClassFourCC** () const  
_ get the class FourCC code |
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Base::MouseRenderDeviceBase::PreloadTextures(const Util::Array<Resources::ResourceId> & texResIds)
load mouse pointer textures

This method must be called to preload texture used by the mouse renderer. The method may be called at any time (also several times).

```cpp
void Base::MouseRenderDeviceBase::UpdatePointers(const Util::Array<CoreGraphics::MousePointer> & pointers)
update the mouse renderer

Update mouse pointers for rendering in the current frame. On some platforms, more then one mouse pointer exists, so this method takes an array of MousePointer objects. Calling this method will replace the previous array of MousePointers.

```cpp
void Base::MouseRenderDeviceBase::RenderPointers()
render mouse pointer(s)

This method should render the pointers describes by the last call to UpdatePointers(). Override this method in a derived platform-specific class.

```cpp
int Core::RefCounted::GetRefCount()
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one

Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Base::MultipleRenderTargetBase
Base::MultipleRenderTargetBase Class Reference

#include <multiplerendertargetbase.h>

Inheritance diagram for Base::MultipleRenderTargetBase:
Detailed Description

**Base** class for render targets. A render targets wraps up to 4 color buffers and an optional depth/stencil buffer into a C++ object. The special default render target represents the backbuffer and default depth/stencil surface.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MultipleRenderTargetBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~MultipleRenderTargetBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>AddRenderTarget (const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp;rt)</strong></td>
<td>set rendertarget</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; CoreGraphics::RenderTarget &gt; &amp;</strong></td>
<td><strong>GetRenderTarget (IndexT i) const</strong></td>
</tr>
<tr>
<td><strong>SizeT GetNumRendertargets () const</strong></td>
<td>get number of rendertargets used</td>
</tr>
<tr>
<td>void <strong>BeginPass ()</strong></td>
<td>begin rendering to the render target</td>
</tr>
<tr>
<td>void <strong>BeginBatch (CoreGraphics::BatchType::Code batchType)</strong></td>
<td>begin a batch</td>
</tr>
<tr>
<td>void <strong>EndBatch ()</strong></td>
<td>end current batch</td>
</tr>
<tr>
<td>void <strong>EndPass ()</strong></td>
<td>end current render pass</td>
</tr>
<tr>
<td>void <strong>SetClearColor (IndexT i, const Math::float4 &amp;color)</strong></td>
<td>set clear color of rendertarget at index</td>
</tr>
<tr>
<td><strong>SetClearColor (float d)</strong></td>
<td>set clear depth</td>
</tr>
<tr>
<td><strong>GetClearColor () const</strong></td>
<td>get clear depth</td>
</tr>
<tr>
<td><strong>SetClearDepth (float d)</strong></td>
<td>set clear depth</td>
</tr>
<tr>
<td><strong>GetClearDepth () const</strong></td>
<td>get clear depth</td>
</tr>
<tr>
<td><strong>SetClearStencil (uchar s)</strong></td>
<td>set clear stencil value</td>
</tr>
<tr>
<td><strong>GetClearStencil () const</strong></td>
<td>get clear stencil value</td>
</tr>
<tr>
<td><strong>uchar GetClearStencil () const</strong></td>
<td>get clear stencil value</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>

The table lists various functions and their descriptions, including methods for counting references, incrementing and releasing references, checking instance of classes, and retrieving class names and FourCC codes.
<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Attributes

<table>
<thead>
<tr>
<th>static const IndexT</th>
<th>MaxNumRenderTargets = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>max number of rendertargets</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Base::ParticleRendererBase
Base::ParticleRendererBase Class Reference

#include <particlerendererbase.h>

Inheritance diagram for Base::ParticleRendererBase:
Detailed Description

**Base** class for platform-specific particle renders.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ParticleRendererBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~ParticleRendererBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <strong>Setup ()</strong></td>
<td>setup the particle renderer</td>
</tr>
<tr>
<td>virtual void <strong>Discard ()</strong></td>
<td>discard the particle renderer</td>
</tr>
<tr>
<td>bool <strong>IsValid ()</strong> const</td>
<td>return true if particle renderer has been setup</td>
</tr>
<tr>
<td>virtual void <strong>BeginAttach ()</strong></td>
<td>begin adding visible particle systems</td>
</tr>
<tr>
<td>void <strong>AddVisibleParticleSystem (const Ptr&lt; Particles::ParticleSystemInstance &gt; &amp;particleSystemInstance)</strong></td>
<td>attach a visible particle system instance</td>
</tr>
<tr>
<td>bool <strong>IsInAttach ()</strong> const</td>
<td>is renderer in attach?</td>
</tr>
<tr>
<td>virtual void <strong>EndAttach ()</strong></td>
<td>finish adding visible particle systems</td>
</tr>
<tr>
<td>void <strong>RenderParticleSystem (const Ptr&lt; Particles::ParticleSystemInstance &gt; &amp;particleSystemInstance)</strong></td>
<td>render particles of previously attached particle system</td>
</tr>
<tr>
<td>int <strong>GetRefCount ()</strong> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Rtti &amp;rtti)</strong> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::String &amp;className)</strong></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>const</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
This method should be called as the very last before an application exits.
Base::ParticleSystemInstanceBase
Base::ParticleSystemInstanceBase
Class Reference

#include <particlesysteminstancebase.h>
Detailed Description

The per-instance object of a ParticleSystem. This is where actual particles are created and updated.

(C) 2008 Radon Labs GmbH
Base::RenderDeviceBase
Base::RenderDeviceBase Class Reference

#include <renderdevicebase.h>

Inheritance diagram for Base::RenderDeviceBase:
Detailed Description

The central rendering object of the Nebula3 core graphics system. This is basically an encapsulation of the Direct3D device. The render device will presents its backbuffer to the display managed by the CoreGraphics::DisplayDevice singleton.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RenderDeviceBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~RenderDeviceBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>SetOverrideDefaultRenderTarget (const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp;rt)</strong></td>
<td>set an override for the default render target (call l Open()!)</td>
</tr>
<tr>
<td>bool <strong>Open ()</strong></td>
<td>open the device</td>
</tr>
<tr>
<td>void <strong>Close ()</strong></td>
<td>close the device</td>
</tr>
<tr>
<td>bool <strong>IsOpen () const</strong></td>
<td>return true if currently open</td>
</tr>
<tr>
<td>void <strong>AttachEventHandler (const Ptr&lt; CoreGraphics::RenderEventHandler &amp;h)</strong></td>
<td>attach a render event handler</td>
</tr>
<tr>
<td>void <strong>RemoveEventHandler (const Ptr&lt; CoreGraphics::RenderEventHandler &amp;h)</strong></td>
<td>remove a render event handler</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; GetDefaultRenderTarget () const</strong></td>
<td>get default render target</td>
</tr>
<tr>
<td>bool <strong>HasPassRenderTarget () const</strong></td>
<td>is a pass rendertarget set?</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; GetPassRenderTarget () const</strong></td>
<td>get current set pass render target</td>
</tr>
<tr>
<td>bool <strong>BeginFrame ()</strong></td>
<td>begin complete frame</td>
</tr>
<tr>
<td>void <strong>BeginPass (constPtr&lt; CoreGraphics::RenderTarget &gt; &amp;rt)</strong></td>
<td>begin renderpass</td>
</tr>
<tr>
<td>void <strong>BeginPass (constPtr&lt; CoreGraphics::ShaderInstance &gt;</strong></td>
<td>renderpass with shader instance</td>
</tr>
</tbody>
</table>

### CoreGraphics

**RenderTarget**

**ShaderInstance**

**RenderEventHandler**
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void BeginPass (const Ptr&lt;CoreGraphics::MultipleRenderTarget &amp;mrt, const Ptr&lt;CoreGraphics::ShaderInstance &amp;passShader)</td>
<td>begin rendering a frame pass with a multiple render target</td>
</tr>
<tr>
<td>void BeginBatch (CoreGraphics::BatchType::Code batchType, const Ptr&lt;CoreGraphics::ShaderInstance &amp;batchShader)</td>
<td>begin rendering a batch inside</td>
</tr>
<tr>
<td>void SetStreamSource (IndexT streamIndex, const Ptr&lt;CoreGraphics::VertexBuffer &amp;vb, IndexT offsetVertexIndex)</td>
<td>set the current vertex stream source</td>
</tr>
<tr>
<td>const Ptr&lt;CoreGraphics::VertexBuffer &amp; &amp;vb</td>
<td>GetStreamVertexBuffer (IndexT streamIndex) const</td>
</tr>
<tr>
<td>IndexT</td>
<td>get currently set vertex buffer</td>
</tr>
<tr>
<td>void SetStreamVertexBuffer (IndexT streamIndex) const</td>
<td>get currently set vertex stream offset</td>
</tr>
<tr>
<td>void SetVertexLayout (const Ptr&lt;CoreGraphics::VertexLayout &amp;vl)</td>
<td>set current vertex layout</td>
</tr>
<tr>
<td>const Ptr&lt;CoreGraphics::VertexLayout &amp; &amp;vl</td>
<td>GetVertexLayout () const</td>
</tr>
<tr>
<td>void SetIndexBuffer (const Ptr&lt;CoreGraphics::IndexBuffer &amp;ib)</td>
<td>set current index buffer</td>
</tr>
<tr>
<td>const Ptr&lt;CoreGraphics::IndexBuffer &amp; &amp;ib</td>
<td>GetIndexBuffer () const</td>
</tr>
<tr>
<td>void SetPrimitiveGroup (const CoreGraphics::PrimitiveGroup &amp;p)</td>
<td>set current primitive group</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const CoreGraphics::PrimitiveGroup &amp; GetPrimitiveGroup () const</code></td>
<td>get current primitive group</td>
</tr>
<tr>
<td><code>void Draw ()</code></td>
<td>draw current primitives</td>
</tr>
<tr>
<td><code>void DrawIndexedInstanced (SizeT numInstances)</code></td>
<td>draw indexed, instanced primitives</td>
</tr>
<tr>
<td><code>void EndBatch ()</code></td>
<td>end current batch</td>
</tr>
<tr>
<td><code>void EndPass ()</code></td>
<td>end current pass</td>
</tr>
<tr>
<td><code>void EndFrame ()</code></td>
<td>end current frame</td>
</tr>
<tr>
<td><code>bool IsInBeginFrame () const</code></td>
<td>check if inside BeginFrame</td>
</tr>
<tr>
<td><code>void Present ()</code></td>
<td>present the rendered scene</td>
</tr>
<tr>
<td><code>CoreGraphics::ImageFileFormat::Code SaveScreenshot (CoreGraphics::ImageFileFormat::Code fmt, const Ptr&lt; IO::Stream &gt; &amp;outStream)</code></td>
<td>save a screenshot to the provided stream</td>
</tr>
<tr>
<td><code>bool GetVisualizeMipMaps () const</code></td>
<td>get visualization of mipmaps flag</td>
</tr>
<tr>
<td><code>void SetVisualizeMipMaps (bool val)</code></td>
<td>set visualization of mipmaps flag</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName)</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static bool <strong>CanCreate</strong> ()</td>
<td>test if a compatible render device can be created on this machine</td>
</tr>
<tr>
<td>static void <strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Static Public Attributes

<table>
<thead>
<tr>
<th>static const IndexT</th>
<th><strong>MaxNumVertexStreams</strong> = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>max number of vertex streams</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
bool NotifyEventArgs (const CoreGraphics::RenderEvent &e)

// notify event handlers about an event
```
Member Function Documentation

bool Base::RenderDeviceBase::CanCreate() [static]

test if a compatible render device can be created on this machine

This static method can be used to check whether a RenderDevice object can be created on this machine before actually instantiating the device object (for instance by checking whether the right Direct3D version is installed). Use this method at application startup to check if the application should run at all.

Reimplemented in Direct3D9::D3D9RenderDevice.

void Base::RenderDeviceBase::SetOverrideDefaultRenderTarget(const Ptr<CoreGraphics::RenderTarget> &rt)

set an override for the default render target (call before Open()!)

Override the default render target (which is normally created in Open()) with a render target provided by the application, this is normally only useful for debugging and testing purposes.

void Base::RenderDeviceBase::AttachEventHandler(const Ptr<CoreGraphics::RenderEventHandler> &h)

attach a render event handler

Attach an event handler to the render device.

void Base::RenderDeviceBase::RemoveEventHandler(const Ptr<CoreGraphics::RenderEventHandler> &h)

remove a render event handler

Remove an event handler from the display device.
bool Base::RenderDeviceBase::NotifyEventHandlers (const CoreGraphics::RenderEvent e) [protected]

notify event handlers about an event

Notify all event handlers about an event.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC code of the object.
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Base::RenderTargetBase
Base::RenderTargetBase Class Reference

#include <rendertargetbase.h>

Inheritance diagram for Base::RenderTargetBase:
Detailed Description

**Base** class for render targets. A render targets wraps up to 4 color buffers and an optional depth/stencil buffer into a C++ object. The special default render target represents the backbuffer and default depth/stencil surface.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>enum</th>
<th>ClearFlag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>clear flags</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RenderTargetBase ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~RenderTargetBase ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>bool IsValid () const</code></td>
<td>return true if valid (has been setup)</td>
</tr>
<tr>
<td><code>void SetDefaultRenderTarget (bool b)</code></td>
<td>set to true if default render target (only called by RenderDevice)</td>
</tr>
<tr>
<td><code>bool IsDefaultRenderTarget () const</code></td>
<td>get default render target flag</td>
</tr>
<tr>
<td><code>void SetWidth (SizeT w)</code></td>
<td>set render target width</td>
</tr>
<tr>
<td><code>SizeT GetWidth () const</code></td>
<td>get width of render target in pixels</td>
</tr>
<tr>
<td><code>void SetHeight (SizeT h)</code></td>
<td>set render target height</td>
</tr>
<tr>
<td><code>SizeT GetHeight () const</code></td>
<td>get height of render target in pixels</td>
</tr>
<tr>
<td><code>void SetAntiAliasQuality (CoreGraphics::AntiAliasQuality::Code c)</code></td>
<td>set antialias quality</td>
</tr>
<tr>
<td><code>CoreGraphics::AntiAliasQuality::Code GetAntiAliasQuality () const</code></td>
<td>get anti-alias-quality</td>
</tr>
<tr>
<td><code>void SetColorBufferFormat</code></td>
<td>add a color buffer</td>
</tr>
<tr>
<td><code>CoreGraphics::PixelFormat::Code GetColorBufferFormat () const</code></td>
<td>get color buffer format at index</td>
</tr>
<tr>
<td><code>void AddDepthStencilBuffer ()</code></td>
<td>internally create a depth/stencil buffer</td>
</tr>
</tbody>
</table>
void AddSharedDepthStencilBuffer (const Ptr< CoreGraphics::RenderTarget >&rt)  
use external depth-stencil buffer

bool HasDepthStencilBuffer () const  
return true if the render target has a depth/stencil buffer

void SetMipMapsEnabled (bool b)  
enable mipmap generation for this render target

bool AreMipMapsEnabled () const  
get mipmap generation flag

void SetResolveTextureResourceId (const Resources::ResourceId &resId)  
set resolve texture resource id

const Resources::ResourceId & GetResolveTextureResourceId () const  
get resolve texture resource id

void SetResolveDepthTextureResourceId (const Resources::ResourceId &resId)  
set optional resolve depth texture resource id

const Resources::ResourceId & GetResolveDepthTextureResourceId () const  
get optional resolve depth texture resource id

void SetResolveTextureWidth (SizeT w)  
set resolve texture width

SizeT GetResolveTextureWidth () const  
get resolve texture width

void SetResolveTextureHeight (SizeT h)  
set resolve texture height

SizeT GetResolveTextureHeight () const  
get resolve texture height

void SetResolveTargetCpuAccess (bool b)  
set cpu access flag

bool GetResolveTargetCpuAccess () const  
get cpu access flag

void SetMRTIndex (IndexT i)  
set optional MRT (Multiple Render Target) index, default is 0
### GetMRTIndex

GetMRTIndex() const

*get multiple-render-target index*

### SetClearFlags

SetClearFlags(uint clearFlags)

*set clear flags*

### GetClearFlags

GetClearFlags() const

*get clear flags*

### SetClearColor

SetClearColor(const Math::float4 &c)

*set clear color*

### GetClearColor

GetClearColor() const

*get clear color*

### SetClearDepth

SetClearDepth(float d)

*set clear depth*

### GetClearDepth

GetClearDepth() const

*get clear depth*

### SetClearStencil

SetClearStencil(uchar s)

*set clear stencil value*

### GetClearStencil

GetClearStencil() const

*get clear stencil value*

### SetResolveRect

SetResolveRect(const Math::rectangle<int> &r)

*set the current resolve rectangle (in pixels)*

### GetResolveRect

GetResolveRect() const

*get resolve rectangle*

### Setup

Setup()

*setup the render target object*

### Discard

Discard()

*discard the render target object*

### BeginPass

BeginPass()

*begin rendering to the render target*

### BeginBatch

BeginBatch(CoreGraphics::BatchType::Code batchType)

*begin a batch*

### EndBatch

EndBatch()

*end current batch*

### EndPass

EndPass()

*end current render pass*
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void GenerateMipLevels ()</code></td>
<td>generate mipmap levels</td>
</tr>
<tr>
<td><code>bool HasResolveTexture ()</code></td>
<td>return true if resolve texture is valid</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::Texture &gt; &amp; GetResolveTexture ()</code></td>
<td>get the resolve texture as Nebula texture object</td>
</tr>
<tr>
<td><code>bool HasCPUResolveTexture ()</code></td>
<td>return true if cpu access resolve texture is valid</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::Texture &gt; &amp; GetCPUResolveTexture ()</code></td>
<td>get the resolve texture as Nebula texture object</td>
</tr>
<tr>
<td><code>bool HasDepthResolveTexture ()</code></td>
<td>return true if resolve texture is valid</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::Texture &gt; &amp; GetDepthResolveTexture ()</code></td>
<td>get the resolve texture as Nebula texture object</td>
</tr>
<tr>
<td><code>virtual void ResolveDepthBuffer ()</code></td>
<td>resolve depth buffer</td>
</tr>
<tr>
<td><code>SizeT GetMemorySize ()</code></td>
<td>get byte size in memory, implemented in platform specific classes</td>
</tr>
<tr>
<td><code>void SetMemoryOffset (SizeT size)</code></td>
<td>set optional memory offset, not used by all platforms</td>
</tr>
<tr>
<td><code>int GetRefCount ()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>Util::String</strong> &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><code>DumpRefCountingLeaks()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String& Core::RefCounted::GetClassName() const [inline, inherited]
get the class name
Get the class name of the object.

const Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Base::ResourceAllocator
Base::ResourceAllocator Class Reference

#include <resourceallocator.h>
Detailed Description

**Base** class of platform-specific resource allocators. A resource allocator provides custom memory allocation from a fixed area of platform specific resource memory. This is the base of a streaming resource management system where resource are streamed in and out of a fixed memory area. The resource manager is free to move the allocated memory blocks around (the addresses in the lumps will be adjusted accordingly and the lump will be notified).

NOTE: The **ResourceAllocator** system performs some operations per-block, thus it's good practice to choose fewer big blocks, instead of many small blocks!

(C) 2008 Radon Labs GmbH
Base::ResourceBase
#include <resourcebase.h>

Inheritance diagram for Base::ResourceBase:
Detailed Description

**Base** class for all **CoreGraphics** resource classes.

(C) 2007 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Usage</th>
<th>resource usage flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>State</td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ResourceBase ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~ResourceBase ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>void SetUsage (Usage usage)</code></td>
<td>set resource usage type</td>
</tr>
<tr>
<td><code>Usage GetUsage () const</code></td>
<td>get resource usage type</td>
</tr>
<tr>
<td><code>void SetAccess (Access access)</code></td>
<td>set resource cpu access type</td>
</tr>
<tr>
<td><code>Access GetAccess () const</code></td>
<td>get cpu access type</td>
</tr>
<tr>
<td><code>void SetAsyncEnabled (bool b)</code></td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td><code>bool IsAsyncEnabled () const</code></td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>void Lock ()</code></td>
<td>set locked to true</td>
</tr>
<tr>
<td><code>void Unlock ()</code></td>
<td>set locked to false</td>
</tr>
<tr>
<td><code>bool IsLocked () const</code></td>
<td>returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td><code>void SetResourceld (const Resourceld &amp;id)</code></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><code>const Resourceld &amp; GetResourceld () const</code></td>
<td>get the resource identifier</td>
</tr>
<tr>
<td><code>void SetLoader (const Ptr&lt; ResourceLoader &gt; &amp;loader)</code></td>
<td>set optional resource loader</td>
</tr>
<tr>
<td><code>const Ptr&lt; ResourceLoader &gt; &amp; GetLoader () const</code></td>
<td>get optional resource loader</td>
</tr>
<tr>
<td><code>void SetSaver (const Ptr&lt; ResourceSaver &gt;</code></td>
<td>set optional resource saver</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const Ptr&lt;ResourceSaver&gt; &amp; GetSaver()</code></td>
<td>get optional resource saver</td>
</tr>
<tr>
<td><code>SizeT GetUseCount()</code></td>
<td>get current use count</td>
</tr>
<tr>
<td><code>virtual State Load()</code></td>
<td>load the resource</td>
</tr>
<tr>
<td><code>virtual void Unload()</code></td>
<td>unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td><code>void SetState(State s)</code></td>
<td>set current state (usually only called during <code>Load()</code>)</td>
</tr>
<tr>
<td><code>State GetState()</code></td>
<td>get current state</td>
</tr>
<tr>
<td><code>bool IsLoaded()</code></td>
<td>return true if current state is Loaded</td>
</tr>
<tr>
<td><code>bool IsPending()</code></td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td><code>bool LoadFailed()</code></td>
<td>return true if current state is Failed</td>
</tr>
<tr>
<td><code>virtual bool Save()</code></td>
<td>save the resource</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by FourCC</td>
</tr>
<tr>
<td>fourcc</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const</td>
<td><strong>Util::String</strong> &amp; <strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
</table>

*dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)*
## Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>IncrUseCount</strong> ()</td>
<td>increment use count</td>
</tr>
<tr>
<td>void</td>
<td><strong>DecrUseCount</strong> ()</td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Resource::State**

**Resources::Resource::Load** ( ) [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the **IsPending()** method will return true as long as the load is in progress, and **IsLoaded()** will become true when the loading process has finished. If the load has failed, **IsPending()** will switch to false and **IsLoaded()** will not be true.

**void**

**Resources::Resource::Unload** ( ) [virtual, inherited]

unload the resource, or cancel the pending load

This will unload the resource. Only call the method when **IsLoaded()** return true. To cancel a pending asynchronous loading process, call the CancelPendingLoad() method.

Reimplemented in **CoreAnimation::AnimResource**, **Base::MeshBase**, **Base::VertexBufferBase**, **Direct3D9::D3D9Shader**, **Direct3D9::D3D9Texture**, **Win360::D3D9IndexBuffer**, **Win360::D3D9VertexBuffer**, and **Models::Model**.

**bool**

**Resources::Resource::Save** ( ) [virtual, inherited]

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Base::ResourceLump
Base::ResourceLump Class Reference

#include <resourcelump.h>
Detailed Description

A resource lump points to a piece of memory from a ResourceAllocator heap.

(C) 2008 Radon Labs GmbH
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**Base::ShaderBase**
Base::ShaderBase Class Reference

#include <shaderbase.h>

Inheritance diagram for Base::ShaderBase:

- Core::RefCounted
- Resources::Resource
- Base::ShaderBase
- Direct3D9::D3D9Shader
- CoreGraphics::Shader
Detailed Description

A shader object groups render states required to render a piece of geometry. Shader objects are not used for rendering directly, instead ShaderInstances are created from a shader.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>resource states <em>(DO NOT CHANGE ORDER!)</em></td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor</td>
<td><code>ShaderBase()</code></td>
<td>virtual destructor</td>
</tr>
<tr>
<td>Destructor</td>
<td><code>~ShaderBase()</code></td>
<td></td>
</tr>
<tr>
<td>Member Function</td>
<td><code>Ptr&lt;CoreGraphics::ShaderInstance&gt;</code></td>
<td>CreateShaderInstance</td>
</tr>
<tr>
<td>Member Function</td>
<td><code>void</code></td>
<td>DiscardShaderInstance</td>
</tr>
<tr>
<td>Member Function</td>
<td><code>const Util::Array&lt;Ptr&lt;CoreGraphics::ShaderInstance&gt;&gt; &amp;inst)</code></td>
<td>GetAllShaderInstances</td>
</tr>
<tr>
<td>Member Function</td>
<td><code>void</code></td>
<td>SetAsyncEnabled</td>
</tr>
<tr>
<td>Member Function</td>
<td><code>bool</code></td>
<td>IsAsyncEnabled</td>
</tr>
<tr>
<td>Member Function</td>
<td><code>void</code></td>
<td>Lock</td>
</tr>
<tr>
<td>Member Function</td>
<td><code>void</code></td>
<td>Unlock</td>
</tr>
<tr>
<td>Member Function</td>
<td><code>bool</code></td>
<td>IsLocked</td>
</tr>
<tr>
<td>Member Function</td>
<td><code>void</code></td>
<td>SetResourceId</td>
</tr>
<tr>
<td>Member Function</td>
<td><code>const Resourceld &amp;</code></td>
<td>GetResourceId</td>
</tr>
<tr>
<td>Member Function</td>
<td><code>void</code></td>
<td>SetLoader</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>const Ptr&lt; ResourceLoader &gt; &amp; GetLoader()</code></td>
<td>get optional resource loader</td>
<td></td>
</tr>
<tr>
<td><code>void SetSaver (const Ptr&lt; ResourceSaver &gt; &amp; saver)</code></td>
<td>set optional resource saver</td>
<td></td>
</tr>
<tr>
<td><code>const Ptr&lt; ResourceSaver &gt; &amp; GetSaver()</code></td>
<td>get optional resource saver</td>
<td></td>
</tr>
<tr>
<td><code>SizeT GetUseCount()</code></td>
<td>get current use count</td>
<td></td>
</tr>
<tr>
<td><code>virtual State Load()</code></td>
<td>load the resource</td>
<td></td>
</tr>
<tr>
<td><code>virtual void Unload()</code></td>
<td>unload the resource, or cancel the load</td>
<td></td>
</tr>
<tr>
<td><code>void SetState (State s)</code></td>
<td>set current state (usually only called during <code>Load()</code>)</td>
<td></td>
</tr>
<tr>
<td><code>State GetState()</code></td>
<td>get current state</td>
<td></td>
</tr>
<tr>
<td><code>bool IsLoaded()</code></td>
<td>return true if current state is Loaded</td>
<td></td>
</tr>
<tr>
<td><code>bool IsPending()</code></td>
<td>return true if current state is Pending</td>
<td></td>
</tr>
<tr>
<td><code>bool LoadFailed()</code></td>
<td>return true if current state is LoadFailed</td>
<td></td>
</tr>
<tr>
<td><code>virtual bool Save()</code></td>
<td>save the resource</td>
<td></td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>get the current refcount</td>
<td></td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
<td></td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
<td></td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti const)</code></td>
<td>return true if this object is instance of given class</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
<td></td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
<td></td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
<td></td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th align="left">static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td align="left"><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>IncrUseCount()</code></td>
<td>Increment use count</td>
</tr>
<tr>
<td>void <code>DecrUseCount()</code></td>
<td>Decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Resource::State**

**Resources::Resource::Load**( ) [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the **IsPending()** method will return true as long as the load is in progress, and **IsLoaded()** will become true when the loading process has finished. If the load has failed, **IsPending()** will switch to false and **IsLoaded()** will not be true.

**void**

**Resources::Resource::Unload**( ) [virtual, inherited]

unload the resource, or cancel the pending load

This will unload the resource. Only call the method when **IsLoaded()** return true. To cancel a pending asynchronous loading process, call the **CancelPendingLoad()** method.

Reimplemented in **CoreAnimation::AnimResource**, **Base::MeshBase**, **Base::VertexBufferBase**, **Direct3D9::D3D9Shader**, **Direct3D9::D3D9Texture**, **Win360::D3D9IndexBuffer**, **Win360::D3D9VertexBuffer**, and **Models::Model**.

**bool**

**Resources::Resource::Save**( ) [virtual, inherited]

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**Base::ShaderInstanceBase**
Base::ShaderInstanceBase Class Reference

#include <shaderinstancebase.h>

Inheritance diagram for Base::ShaderInstanceBase:

```
Core::RefCounted

Base::ShaderInstanceBase

Direct3D9::D3D9ShaderInstance

CoreGraphics::ShaderInstance
```
Detailed Description

A shader instance object is created from a shader and contains a local copy of the original shader state which can be modified through ShaderVariable objects. Shader instance objects are created directly through the shader server.

(C) 2007 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShaderInstanceBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~ShaderInstanceBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>void Discard ()</strong></td>
<td><em>discard the shader instance, must be called when instance no longer needed</em></td>
</tr>
<tr>
<td><strong>bool IsValid () const</strong></td>
<td><em>return true if this object is valid</em></td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::Shader &gt; &amp; GetOriginalShader () const</strong></td>
<td><em>get pointer to original shader which created this instance</em></td>
</tr>
<tr>
<td><strong>bool HasVariableByName (const CoreGraphics::ShaderVariable::Name &amp;n) const</strong></td>
<td><em>return true if the shader instance has a variable by name</em></td>
</tr>
<tr>
<td><strong>bool HasVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &amp;n) const</strong></td>
<td><em>return true if shader has variable by semantic</em></td>
</tr>
<tr>
<td><strong>SizeT GetNumVariables () const</strong></td>
<td><em>get number of variables</em></td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp; GetVariableByIndex (IndexT i) const</strong></td>
<td><em>get a variable by index</em></td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp; GetVariableByName (const CoreGraphics::ShaderVariable::Name &amp;n) const</strong></td>
<td><em>get a variable by name</em></td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp; GetVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &amp;s) const</strong></td>
<td><em>get a variable by semantic</em></td>
</tr>
<tr>
<td><strong>bool HasVariation (CoreGraphics::ShaderFeature::Mask featureMask) const</strong></td>
<td></td>
</tr>
</tbody>
</table>
SizeT GetNumVariations() const
get number of variations in the shader

const Ptr<CoreGraphics::ShaderVariation> & GetVariationByIndex(IndexT i) const
get shader variation by index

const Ptr<CoreGraphics::ShaderVariation> & GetVariationByFeatureMask(CoreGraphics::ShaderFeature::Mask featureMask) const
get shader variation by feature mask

bool SelectActiveVariation(CoreGraphics::ShaderFeature::Mask featureMask)
select active variation by feature mask, return true if active variation has been changed

const Ptr<CoreGraphics::ShaderVariation> & GetActiveVariation() const
get currently active variation

SizeT Begin()
begin rendering through the currently selected variation, returns no. passes

void BeginPass(IndexT passIndex)
begin pass

void Commit()
commit changes before rendering

void EndPass()
end pass

void End()
end rendering through variation

int GetRefCount() const
get the current refcount

void AddRef()
increment refcount by one

void Release()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf(const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf(const Util::String &className) const
return true if this object is instance of given class
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, by fourcc</td>
</tr>
<tr>
<td><code>Util::String &amp; GetClassName()</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC()</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
</table>

*dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)*
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void</td>
<td><strong>Setup</strong> (const <code>Ptr&lt; CoreGraphics::Shader &gt;</code> &amp;origShader)</td>
</tr>
<tr>
<td></td>
<td><em>setup the shader instance from its original shader object</em></td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Cleanup</strong> ()</td>
</tr>
<tr>
<td></td>
<td><em>discard the shader instance</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

void Base::ShaderInstanceBase::Discard()

discard the shader instance, must be called when instance no longer needed

This method must be called when the object is no longer needed for proper cleanup.

void Base::ShaderInstanceBase::Setup(const Ptr<CoreGraphics::Shader> origShader) [protected, virtual]

setup the shader instance from its original shader object

Override this method in an API-specific subclass to setup the shader instance, and call the parent class for proper setup.

Reimplemented in Direct3D9::D3D9ShaderInstance.

void Base::ShaderInstanceBase::Cleanup() [protected, virtual]

discard the shader instance

Override this method in an API-specific subclass to undo the setup in OnInstantiate(), then call parent class to finalize the cleanup.

Reimplemented in Direct3D9::D3D9ShaderInstance.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one

Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Base::ShaderServerBase
Base::ShaderServerBase Class Reference

#include <shaderserverbase.h>

Inheritance diagram for Base::ShaderServerBase:
Detailed Description

In Nebula3, all shaders required by an application are loaded at once by the central ShaderServer. The shader server loads all shaders in ShaderServer::Open() from the location defined by the "shaders:" assign.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShaderServerBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~ShaderServerBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>bool Open ()</strong></td>
<td>Open the shader server</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>Close the shader server</td>
</tr>
<tr>
<td><strong>bool IsOpen () const</strong></td>
<td>Return true if the shader server is open</td>
</tr>
<tr>
<td><strong>bool HasShader (const Resources::ResourceId &amp;resId)</strong></td>
<td>Return true if a shader exists</td>
</tr>
<tr>
<td><strong>Ptr&lt; CoreGraphics::ShaderInstance &gt; CreateShaderInstance (const Resources::ResourceId &amp;resId)</strong></td>
<td>Create a new shader instance</td>
</tr>
<tr>
<td><strong>const Util::Dictionary&lt; Resources::ResourceId, Ptr&lt; CoreGraphics::Shader &gt; &gt; &amp; GetAllShaders () const</strong></td>
<td>Get all loaded shaders</td>
</tr>
<tr>
<td><strong>void SetActiveShaderInstance (const CoreGraphics::ShaderInstance &amp;shaderInst)</strong></td>
<td>Set currently active shader instance</td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp; GetActiveShaderInstance () const</strong></td>
<td>Get currently active shader instance</td>
</tr>
<tr>
<td><strong>void ResetFeatureBits ()</strong></td>
<td>Reset the current feature bits</td>
</tr>
<tr>
<td><strong>void SetFeatureBits (CoreGraphics::ShaderFeature::Mask)</strong></td>
<td>Set shader feature by bit mask</td>
</tr>
<tr>
<td><strong>void ClearFeatureBits (CoreGraphics::ShaderFeature::Mask)</strong></td>
<td>Clear shader feature by bit mask</td>
</tr>
<tr>
<td><strong>CoreGraphics::ShaderFeature::Mask GetFeatureBits () const</strong></td>
<td>Get shader feature by bit mask</td>
</tr>
</tbody>
</table>
get the current feature mask

CoreGraphics::ShaderFeature::Mask FeatureStringToMask (const Util::String &str)
convert a shader feature string into a feature bit mask

Util::String FeatureMaskToString (CoreGraphics::ShaderFeature::Mask mask)
convert shader feature bit mask into string

void ApplyObjectId (IndexT i)
apply an object id

bool HasSharedVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &sem) const
return true if a shared variable exists by semantic

SizeT GetNumSharedVariables () const
get number of shared variables

cnst Ptr< CoreGraphics::ShaderVariable > & GetSharedVariableByIndex (IndexT i) const
get a shared variable by index

cnst Ptr< CoreGraphics::ShaderVariable > & GetSharedVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &sem) const
get a shared variable by semantic

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc
<table>
<thead>
<tr>
<th>bool IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool IsA (const Util::String &amp;rttiName)</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool IsA (const Util::FourCC &amp;rttiFour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp; GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Ptr< ShaderInstance >**
Base::ShaderServerBase::CreateShaderInstance (const Resources::ResourceId& resId) &

create a new shader instance

This creates a clone of a template shader. This is the only method to create a new shader object. When the shader instance is no longer needed, call UnregisterShaderInstance() for proper cleanup.

**int**
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

**void**
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

**void**
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

**const Util::String &**
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

**Util::FourCC**
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**Base::ShaderVariableBase**
Base::ShaderVariableBase Class Reference

#include <shadervariablebase.h>

Inheritance diagram for Base::ShaderVariableBase:
Detailed Description

Provides direct access to a shader's global variable. The fastest way to change the value of a shader variable is to obtain a pointer to a shader variable once, and use it repeatedly to set new values.

(C) 2006 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enum</code></td>
<td><code>Type</code></td>
</tr>
<tr>
<td></td>
<td><code>shader variable types</code></td>
</tr>
<tr>
<td><code>typedef Util::StringAtom</code></td>
<td><code>Name</code></td>
</tr>
<tr>
<td></td>
<td><code>shader variable name typedef</code></td>
</tr>
<tr>
<td><code>typedef Util::StringAtom</code></td>
<td><code>Semantic</code></td>
</tr>
<tr>
<td></td>
<td><code>shader variable semantic typedef</code></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShaderVariableBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~ShaderVariableBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>Ptr&lt; CoreGraphics::ShaderVariableInstance &gt;</strong></td>
<td><strong>CreateInstance ()</strong> create a shader variable instance</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td><strong>GetType () const</strong> get the data type of the variable</td>
</tr>
<tr>
<td>const <strong>Name &amp;</strong></td>
<td><strong>GetName () const</strong> get the name of the variable</td>
</tr>
<tr>
<td>const <strong>Semantic &amp;</strong></td>
<td><strong>GetSemantic () const</strong> get the semantics of the variable</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetInt</strong> (int value) set int value</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetIntArray</strong> (const int *values, SizeT count) set int array values</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetFloat</strong> (float value) set float value</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetFloatArray</strong> (const float *values, SizeT count) set float array values</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetFloat4</strong> (const Math::float4 &amp;value) set vector value</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetFloat4Array</strong> (const Math::float4 *values, SizeT count) set vector array values</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetMatrix</strong> (const Math::matrix44 &amp;value) set matrix value</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>SetMatrixArray</strong> (const</td>
</tr>
</tbody>
</table>
void Math::matrix44 *values, SizeT count)
  set matrix array values

void SetBool (bool value)
  set bool value

void SetBoolArray (const bool *values, SizeT count)
  set bool array values

void SetTexture (const Ptr<CoreGraphics::Texture> &value)
  set texture value

int GetRefCount () const
  get the current refcount

void AddRef ()
  increment refcount by one

void Release ()
  decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
  return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
  return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
  return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
  return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
  return true if this object is instance of
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool <code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp; <code>GetClassName</code> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC</code> <code>GetClassFourCC</code> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static Util::String</th>
<th><strong>TypeToString</strong> <em>(Type t)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>convert type to string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static Type</th>
<th><strong>StringToType</strong> <em>(const Util::String &amp;str)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>convert string to type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application <em>(NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetType(Type t)</code></td>
<td>set variable type</td>
</tr>
<tr>
<td><code>voidSetName(const Name &amp;n)</code></td>
<td>set variable name</td>
</tr>
<tr>
<td><code>void SetSemantic(const Semantic &amp;s)</code></td>
<td>set variable semantic</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Base::ShaderVariableInstanceBase
Base::ShaderVariableInstanceBase
Class Reference

#include <shadervariableinstancebase.h>

Inheritance diagram for Base::ShaderVariableInstanceBase:
Detailed Description

A ShaderVariableInstance associates a value with a ShaderVariable and can apply that value at any time to the ShaderVariable. Setting the value on a ShaderVariableInstance will just store the value but not change the actual ShaderVariable. Only calling `Apply()` will set the stored value on the ShaderVariable. ShaderVariableInstance objects are used to manage per-instance state when rendering ModelNodeInstances.

NOTE: you cannot set arrays through a ShaderVariableInstance!

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShaderVariableInstanceBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~ShaderVariableInstanceBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>Setup (const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp;var)</strong></td>
<td>Setup the object from a shader variable</td>
</tr>
<tr>
<td><strong>Prepare (CoreGraphics::ShaderVariable::Type)</strong></td>
<td>Prepare the object for late-binding</td>
</tr>
<tr>
<td><strong>Bind (const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp;var)</strong></td>
<td>Late-bind the variable instance to a shader variable</td>
</tr>
<tr>
<td><strong>GetShaderVariable () const</strong></td>
<td>Get the associated shader variable</td>
</tr>
<tr>
<td><strong>Apply ()</strong></td>
<td>Apply local value to shader variable</td>
</tr>
<tr>
<td><strong>SetInt (int value)</strong></td>
<td>Set int value</td>
</tr>
<tr>
<td><strong>SetFloat (float value)</strong></td>
<td>Set float value</td>
</tr>
<tr>
<td><strong>SetFloat4 (const Math::float4 &amp;value)</strong></td>
<td>Set float4 value</td>
</tr>
<tr>
<td><strong>SetMatrix (const Math::matrix44 &amp;value)</strong></td>
<td>Set matrix44 value</td>
</tr>
<tr>
<td><strong>SetBool (bool value)</strong></td>
<td>Set bool value</td>
</tr>
<tr>
<td><strong>SetTexture (const Ptr&lt; CoreGraphics::Texture &gt; &amp;value)</strong></td>
<td>Set texture value</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void SetValue</code> (const <code>Util::Variant</code> &amp;v)</td>
<td>set value directly</td>
</tr>
<tr>
<td><code>int GetRefCount</code> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef</code> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release</code> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf</code> (const <code>Util::String</code> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf</code> (const <code>Util::FourCC</code> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class or a derived class</td>
</tr>
<tr>
<td><code>bool IsA</code> (const <code>Util::String</code> &amp;rttiName) const</td>
<td>return true if this object is instance of given class or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const </code>Util::String` &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

void
Base::ShaderVariableInstanceBase::Apply( )

apply local value to shader variable

Todo:
  : hmm, the dynamic type switch is sort of lame...

int
Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Base::ShaderVariationBase
# Base::ShaderVariationBase Class Reference

```cpp
#include <shadervariationbase.h>
```

Inheritance diagram for Base::ShaderVariationBase:
```plaintext
Core::RefCounted

Base::ShaderVariationBase

Direct3D9::D3D9ShaderVariation

CoreGraphics::ShaderVariation
```
Detailed Description

A shader variation is part of a shader which implements a specific behaviour of the shader, identified by a set of "features". Shader variations may implement a depth-only version of the shader, or geometry-deformed-versions of the shader like skinning or shape-blending. There is no pre-defined set of variation feature, this depends on the actually implemented render pipeline.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Public Member Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShaderVariationBase ()</strong></td>
</tr>
<tr>
<td><em>constructor</em></td>
</tr>
<tr>
<td>virtual <strong>~ShaderVariationBase ()</strong></td>
</tr>
<tr>
<td><em>destructor</em></td>
</tr>
<tr>
<td>const Name &amp; <strong>GetName ()</strong> const</td>
</tr>
<tr>
<td><em>get the shader variation's name</em></td>
</tr>
<tr>
<td>CoreGraphics::ShaderFeature::Mask <strong>GetFeatureMask ()</strong> const</td>
</tr>
<tr>
<td><em>get the feature bit mask of this variation</em></td>
</tr>
<tr>
<td>SizeT <strong>GetNumPasses ()</strong> const</td>
</tr>
<tr>
<td><em>get number of passes in this variation</em></td>
</tr>
<tr>
<td>int <strong>GetRefCount ()</strong> const</td>
</tr>
<tr>
<td><em>get the current refcount</em></td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
</tr>
<tr>
<td><em>increment refcount by one</em></td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
</tr>
<tr>
<td><em>decrement refcount and destroy object if refcount is zero</em></td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td><em>return true if this object is instance of given class</em></td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
</tr>
<tr>
<td><em>return true if this object is instance of given class by string</em></td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td><em>return true if this object is instance of given class by fourcc</em></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td><em>return true if this object is instance of given class, or a derived class</em></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td><em>return true if this object is instance of given class, or a derived class, by string</em></td>
</tr>
<tr>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC)</td>
</tr>
<tr>
<td>bool</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
</tr>
<tr>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
</tr>
<tr>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>SetName</code> (const <code>Name</code> &amp;n)</td>
<td>set variation name</td>
</tr>
<tr>
<td>void <code>SetFeatureMask</code> (CoreGraphics::ShaderFeature::Mask m)</td>
<td>set feature bit mask of this variation</td>
</tr>
<tr>
<td>void <code>SetNumPasses</code> (SizeT n)</td>
<td>set number of passes</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Base::ShapeRendererBase
Base::ShapeRendererBase Class Reference

#include <shaprendererbase.h>

Inheritance diagram for Base::ShapeRendererBase:
Detailed Description

Base class of ShapeRenderer, can render a number of shapes, mainly for debug visualization.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShapeRendererBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~ShapeRendererBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Open ()</strong></td>
<td>open the shape renderer</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>close the shape renderer</td>
</tr>
<tr>
<td><strong>IsOpen () const</strong></td>
<td>return true if open</td>
</tr>
<tr>
<td><strong>DeleteShapesByThreadId</strong> (Threading::ThreadId threadId)**</td>
<td>delete shapes of given thread id</td>
</tr>
<tr>
<td><strong>AddShape (const CoreGraphics::RenderShape &amp;shape)</strong></td>
<td>add a shape for deferred rendering (can be called from outside render loop)</td>
</tr>
<tr>
<td><strong>AddShapes (const Util::Array&lt;CoreGraphics::RenderShape &gt; &amp;shapeArray)</strong></td>
<td>add multiple shapes</td>
</tr>
<tr>
<td><strong>AddWireFrameBox (const Math::bbox &amp;boundingBox, const Math::float4 &amp;color, Threading::ThreadId threadId)</strong></td>
<td>add wireframe box</td>
</tr>
<tr>
<td><strong>DrawShapes ()</strong></td>
<td>draw deferred shapes and clear deferred stack, must be called inside render loop</td>
</tr>
<tr>
<td><strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td></td>
</tr>
</tbody>
</table>
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String& Core::RefCounted::GetClassName() const [inline, inherited]
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Base::SkinnedMeshDrawInfoBase
Base::SkinnedMeshDrawInfoBase
Class Reference

#include <skinnedmeshdrawinfobase.h>

Inheritance diagram for Base::SkinnedMeshDrawInfoBase:
Detailed Description

Contains per-instance information for a skinned-mesh draw operation. Add platform-specific attributes by deriving from this class. This is usually only used by platforms which use software skinning.

(C) 2009 Radon Labs GmbH
Base::SkinnedMeshRendererBase
Base::SkinnedMeshRendererBase
Class Reference

#include <skinnedmeshrendererbase.h>

Inheritance diagram for Base::SkinnedMeshRendererBase:
Detailed Description

Wraps platform-specific rendering of a skinned mesh. For a platform which supports GPU skinning (e.g. anything except the Wii), simply call the `DrawGPUSkinnedMesh()`. For a software-skinned platform, call the `UpdateSoftwareSkinnedMesh()` ideally once per frame (although the method makes sure, that a mesh isn't skinned twice even when the method is called multiple times), and then use the returned `DrawHandle` to call `DrawSoftwareSkinnedMesh()` several times per frame.

See `CharacterNodeInstance` for details.

(C) 2008 Radon Labs GmbH
Public Types

typedef IndexT DrawHandle

an abstract draw handle
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SkinnedMeshRendererBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~SkinnedMeshRendererBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>Setup ()</strong></td>
<td>setup the renderer</td>
</tr>
<tr>
<td>void <strong>Discard ()</strong></td>
<td>discard the renderer</td>
</tr>
<tr>
<td>bool <strong>IsValid ()</strong> const</td>
<td>return true if renderer is valid</td>
</tr>
<tr>
<td><strong>GetSkinningTechnique ()</strong> const</td>
<td>get the skinning technique used by the renderer</td>
</tr>
<tr>
<td>int <strong>GetRefCount ()</strong> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Rtti &amp;rtti)</strong> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Util::String &amp;className)</strong> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA (const Rtti &amp;rtti)</strong> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA (const Util::String &amp;rttiName)</strong></td>
<td>return true if this object is instance of given class,</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void OnBeginFrame ()</strong></td>
<td>call once at beginning of frame</td>
</tr>
<tr>
<td><strong>void OnEndFrame ()</strong></td>
<td>call once at end of frame (after rendering)</td>
</tr>
<tr>
<td><strong>void BeginGatherSkins ()</strong></td>
<td>begin gathering software-skinned meshes</td>
</tr>
<tr>
<td><strong>void RegisterSoftwareSkinnedMesh</strong> (const **Ptr&lt; Characters::CharacterInstance &gt; &amp;charInst, const **Ptr&lt; CoreGraphics::Mesh &gt; &amp;mesh)</td>
<td>update a software skinned mesh</td>
</tr>
<tr>
<td><strong>void EndGatherSkins ()</strong></td>
<td>end gathering software-skinned meshes</td>
</tr>
<tr>
<td><strong>void UpdateSoftwareSkinnedMeshes ()</strong></td>
<td>update software-skinned meshes</td>
</tr>
<tr>
<td><strong>void DrawSoftwareSkinnedMesh</strong> (<strong>DrawHandle h, IndexT primGroupIndex</strong>)</td>
<td>draw a software skinned mesh</td>
</tr>
<tr>
<td><strong>void DrawGPUSkinnedMesh</strong> (const **Ptr&lt; Characters::CharacterInstance &gt; &amp;charInst, const **Ptr&lt; CoreGraphics::Mesh &gt; &amp;mesh, IndexT primGroupIndex, const **Util::Array&lt; IndexT &gt; &amp;jointPalette, const **Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp;jointPaletteShdVar)</td>
<td>draw a hardware skinned mesh</td>
</tr>
<tr>
<td><strong>void DrawGPUTextureSkinnedMesh</strong> (const **Ptr&lt; Characters::CharacterInstance &gt; &amp;charInst, const **Ptr&lt; CoreGraphics::Mesh &gt; &amp;mesh, IndexT primGroupIndex, const **Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp;charInstShaderVar)</td>
<td>draw a skinned mesh</td>
</tr>
<tr>
<td><strong>IndexT AllocJointTextureRow ()</strong></td>
<td>allocate a row index in the joint texture</td>
</tr>
<tr>
<td><strong>void FreeJointTextureRow</strong> (IndexT rowIndex)</td>
<td>free a row index in the joint texture</td>
</tr>
<tr>
<td><strong>void AcquireJointTextureRowPointer</strong> (const **Ptr&lt;</td>
<td></td>
</tr>
<tr>
<td>void *</td>
<td><code>Characters::CharacterInstance</code> &gt; &amp;charInst, SizeT &amp;outRowPitch)</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get a pointer to the joint texture row for the given character instance</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

Characters::SkinningTechnique::Code
Base::SkinnedMeshRendererBase::GetSkinningTechnique( ) const [inline]

get the skinning technique used by the renderer

Override this method in a platform specific subclass!

Reimplemented in Win32::Win32SkinnedMeshRenderer.

SkinnedMeshRendererBase::DrawHandle
Base::SkinnedMeshRendererBase::RegisterSoftwareSkinnedMesh( const Ptr<Characters::CharacterInstance> &
    const Ptr<CoreGraphics::Mesh> &
)

update a software skinned mesh

This method should only be called when RequiresSoftwareSkinning() returns true!

This registers a mesh for software-skinning in the
UpdateSoftwareSkinnedMeshes() which must be called after
EndGatherSkins().

This method may be called more then once per character-instance/mesh combination! The method will drop duplicates.

void
Base::SkinnedMeshRendererBase::UpdateSoftwareSkinnedMeshes( ) [protected]

update software-skinned meshes

On platforms with software-skinning, this method should perform the
skinning for all meshes gathered during the GatherSkins pass.

void
Base::SkinnedMeshRendererBase::DrawSoftwareSkinnedMesh(DrawHandle h,
    IndexT primGroupIndex
draw a software skinned mesh

This method should only be called when RequiresSoftwareSkinning() returns true!

Software-skinning platforms call this method with the DrawHandle returned by UpdateSoftwareSkinnedMesh() to draw a portion of the skinned mesh:

```cpp
void Base::SkinnedMeshRendererBase::DrawGPUSkinnedMesh(
    const Ptr<Characters::CharacterInstance> &charInst,
    const Ptr<CoreGraphics::Mesh> &mesh,
    IndexT primGroupIndex,
    const Util::Array<IndexT> &jointPalette,
    const Ptr<CoreGraphics::ShaderVariable> &jointPaletteShdVar)
```

draw a hardware skinned mesh

This method should only be called when RequiresSoftwareSkinning() returns false!

This is the skinned-mesh rendering method for platforms which do skinning on the GPU.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

void Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Base::StreamTextureSaverBase
Base::StreamTextureSaverBase Class Reference

#include <streamtexturesaverbase.h>

Inheritance diagram for Base::StreamTextureSaverBase:
Detailed Description

Allows to save texture data in a standard file format into a stream.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StreamTextureSaverBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~StreamTextureSaverBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <strong>SetStream</strong> (const Ptr&lt; IO::Stream &gt; &amp;stream)</td>
<td>Set stream to save to</td>
</tr>
<tr>
<td>const Ptr&lt; IO::Stream &gt; &amp; <strong>GetStream</strong> () const</td>
<td>Get save-stream</td>
</tr>
<tr>
<td>void <strong>SetFormat</strong>(CoreGraphics::ImageFileFormat::Code fmt)</td>
<td>Set file format (default is JPG)</td>
</tr>
<tr>
<td><strong>GetFormat</strong> () const</td>
<td>Get file format</td>
</tr>
<tr>
<td>void <strong>SetMipLevel</strong>(IndexT mipLevel)</td>
<td>Set the mip level to save (default is 0, the top level)</td>
</tr>
<tr>
<td><strong>GetMipLevel</strong> () const</td>
<td>Get the mip level to save</td>
</tr>
<tr>
<td>virtual bool <strong>OnSave</strong> ()</td>
<td>Called by resource when a save is requested</td>
</tr>
<tr>
<td>virtual void <strong>OnAttachToResource</strong> (const Ptr&lt; Resource &gt; &amp;res)</td>
<td>Called when the resource saver is attached to its resource</td>
</tr>
<tr>
<td>virtual void <strong>OnRemoveFromResource</strong> ()</td>
<td>Called when the resource saver is removed from its resource</td>
</tr>
<tr>
<td>bool <strong>IsAttachedToResource</strong> () const</td>
<td>Return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; <strong>GetResource</strong> () const</td>
<td>Get pointer to resource</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName() const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC() const</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

  dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Base::SystemInfoBase
#include <systeminfobase.h>

Inheritance diagram for Base::SystemInfoBase:
Detailed Description

Provide runtime-system-information.

(C) 2008 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>Platform</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>host platforms</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>CpuType</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>CPU types</em></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Class</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>SystemInfoBase</strong>()</td>
<td>constructor</td>
</tr>
<tr>
<td></td>
<td><strong>GetPlatform</strong>() const</td>
<td>get host platform</td>
</tr>
<tr>
<td>Platform</td>
<td><strong>GetCpuType</strong>() const</td>
<td>get cpu type</td>
</tr>
<tr>
<td>CpuType</td>
<td><strong>GetNumCpuCores</strong>() const</td>
<td>get number of processors</td>
</tr>
<tr>
<td></td>
<td><strong>GetPageSize</strong>() const</td>
<td>get page size</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PlatformAsString</code></td>
<td><code>static Util::String PlatformAsString (Platform p)</code></td>
<td>convert platform to string</td>
</tr>
<tr>
<td><code>CpuTypeAsString</code></td>
<td><code>static Util::String CpuTypeAsString (CpuType cpu)</code></td>
<td>convert CpuType to string</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**Base::TextRendererBase**
Base::TextRendererBase Class Reference

#include <textrendererbase.h>

Inheritance diagram for Base::TextRendererBase:

- Core::RefCounted
- Base::TextRendererBase
- Direct3D9::D3D9TextRenderer
- CoreGraphics::TextRenderer
Detailed Description

**Base** class for text rendering (don't use this for high-quality text rendering).

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TextRendererBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~TextRendererBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Open ()</strong></td>
<td>open the text renderer</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>close the text renderer</td>
</tr>
<tr>
<td><strong>IsOpen () const</strong></td>
<td>check if text renderer open</td>
</tr>
<tr>
<td><strong>DrawTextElements ()</strong></td>
<td>draw the accumulated text</td>
</tr>
<tr>
<td><strong>DeleteTextElementsByThreadId</strong></td>
<td>delete added text by thread id</td>
</tr>
<tr>
<td><strong>AddTextElement (const CoreGraphics::TextElement &amp;textElement)</strong></td>
<td>add text element for rendering</td>
</tr>
<tr>
<td><strong>AddTextElements (const Util::Array&lt;CoreGraphics::TextElement &gt; &amp;textElement)</strong></td>
<td>add multiple text elements for rendering</td>
</tr>
<tr>
<td><strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::String &amp;className)</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong></td>
<td>return true if this object is instance of given class by FourCC</td>
</tr>
<tr>
<td>bool</td>
<td>const</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>return true if this object is instance of given class by fourcc</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
get the class name
Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
```
builds only!

This method should be called as the very last before an application exits.
Base::TextureBase
#include <texturebase.h>

Inheritance diagram for Base::TextureBase:

```
Base::TextureBase
  ↓
Base::ResourceBase
  ↓
Resources::Resource
  ↓
Core::RefCounted
  ↓
Direct3D9::D3D9Texture
  ↓
CoreGraphics::Texture
```
Detailed Description

The base class for texture objects.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>Enum</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Type</strong></td>
<td>texture types</td>
</tr>
<tr>
<td></td>
<td><strong>CubeFace</strong></td>
<td>cube map face</td>
</tr>
<tr>
<td></td>
<td><strong>Usage</strong></td>
<td>resource usage flags</td>
</tr>
<tr>
<td></td>
<td><strong>State</strong></td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enum</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Type</strong></td>
<td>texture types</td>
</tr>
<tr>
<td></td>
<td><strong>CubeFace</strong></td>
<td>cube map face</td>
</tr>
<tr>
<td></td>
<td><strong>Usage</strong></td>
<td>resource usage flags</td>
</tr>
<tr>
<td></td>
<td><strong>State</strong></td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TextureBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~TextureBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>GetType () const</strong></td>
<td>Get texture type</td>
</tr>
<tr>
<td><strong>GetWidth () const</strong></td>
<td>Get width of texture</td>
</tr>
<tr>
<td><strong>GetHeight () const</strong></td>
<td>Get height of texture (if 2d or 3d texture)</td>
</tr>
<tr>
<td><strong>GetDepth () const</strong></td>
<td>Get depth of texture (if 3d texture)</td>
</tr>
<tr>
<td><strong>GetNumMipLevels () const</strong></td>
<td>Get number of mip levels</td>
</tr>
<tr>
<td><strong>GetSkippedMips () const</strong></td>
<td>Get number of currently skipped mip levels</td>
</tr>
<tr>
<td><strong>SetSkippedMips (SizeT m)</strong></td>
<td>Set number of currently skipped mip levels</td>
</tr>
<tr>
<td><strong>GetPixelFormat () const</strong></td>
<td>Get pixel format of the texture</td>
</tr>
<tr>
<td><strong>Map (IndexT mipLevel, Access accessMode, MapInfo &amp;outMapInfo)</strong></td>
<td>Map the a texture mip level for CPU access</td>
</tr>
<tr>
<td><strong>Unmap (IndexT mipLevel)</strong></td>
<td>Unmap texture after CPU access</td>
</tr>
<tr>
<td><strong>MapCubeFace (CubeFace face, IndexT mipLevel, Access accessMode, MapInfo &amp;outMapInfo)</strong></td>
<td>Map a cube map face for CPU access</td>
</tr>
<tr>
<td><strong>UnmapCubeFace (CubeFace face, IndexT mipLevel)</strong></td>
<td>Unmap cube map face after CPU access</td>
</tr>
<tr>
<td><strong>SetUsage (Usage usage)</strong></td>
<td>Set usage</td>
</tr>
</tbody>
</table>
set resource usage type

Usage
- **GetUsage () const**
  - get resource usage type

void
- **SetAccess (Access access)**
  - set resource cpu access type

Access
- **GetAccess () const**
  - get cpu access type

void
- **SetAsyncEnabled (bool b)**
  - request synchronous/asynchronous resource loading

bool
- **IsAsyncEnabled () const**
  - return true if asynchronous resource loading is enabled

void
- **Lock ()**
  - set locked to true

void
- **Unlock ()**
  - set locked to false

bool
- **IsLocked () const**
  - returns true if resource will be used as source for copy process soon

void
- **SetResourceId (const ResourceId &id)**
  - set the resource identifier

const ResourceId &
- **GetResourceId () const**
  - get the resource identifier

void
- **SetLoader (const Ptr<ResourceLoader> &loader)**
  - set optional resource loader

const Ptr<ResourceLoader> &
- **GetLoader () const**
  - get optional resource loader

void
- **SetSaver (const Ptr<ResourceSaver> &saver)**
  - set optional resource saver

const Ptr<ResourceSaver> &
- **GetSaver () const**
  - get optional resource saver

SizeT
- **GetUseCount () const**
  - get current use count

virtual State
- **Load ()**
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>Unload ()</strong></td>
<td>unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td>void <strong>setState (State s)</strong></td>
<td>set current state (usually only called during <strong>Load()</strong>!)</td>
</tr>
<tr>
<td><strong>getState () const</strong></td>
<td>get current state</td>
</tr>
<tr>
<td>bool <strong>IsLoaded () const</strong></td>
<td>return true if current state is Loaded</td>
</tr>
<tr>
<td>bool <strong>IsPending () const</strong></td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td>bool <strong>LoadFailed () const</strong></td>
<td>return true if current state is Failed</td>
</tr>
<tr>
<td>virtual bool <strong>Save ()</strong></td>
<td>save the resource</td>
</tr>
<tr>
<td>int <strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Rtti &amp;rtti)</strong> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Util::String &amp;className)</strong> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA (const Rtti &amp;rtti)</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA (const Util::String &amp;rttiName)</strong> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
# Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetType (Type t)</code></td>
<td>set texture type</td>
</tr>
<tr>
<td><code>void SetWidth (SizeT w)</code></td>
<td>set texture width</td>
</tr>
<tr>
<td><code>void SetHeight (SizeT h)</code></td>
<td>set texture height</td>
</tr>
<tr>
<td><code>void SetDepth (SizeT d)</code></td>
<td>set texture depth</td>
</tr>
<tr>
<td><code>void SetNumMipLevels (SizeT n)</code></td>
<td>set number of mip levels</td>
</tr>
<tr>
<td><code>void SetPixelFormat (CoreGraphics::PixelFormat::Code f)</code></td>
<td>set pixel format</td>
</tr>
<tr>
<td><code>void IncrUseCount ()</code></td>
<td>increment use count</td>
</tr>
<tr>
<td><code>void DecrUseCount ()</code></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>MapInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>access info filled by Map methods</td>
<td>More...</td>
</tr>
</tbody>
</table>
Member Function Documentation

`Resource::State`  
`Resources::Resource::Load()` [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the `IsPending()` method will return true as long as the load is in progress, and `IsLoaded()` will become true when the loading process has finished. If the load has failed, `IsPending()` will switch to false and `IsLoaded()` will not be true.

`void`  
`Resources::Resource::Unload()` [virtual, inherited]

unload the resource, or cancel the pending load

This will unload the resource. Only call the method when `IsLoaded()` return true. To cancel a pending asynchronous loading process, call the `CancelPendingLoad()` method.

Reimplemented in `CoreAnimation::AnimResource`,  
`Base::MeshBase`, `Base::VertexBufferBase`,  
`Direct3D9::D3D9Shader`, `Direct3D9::D3D9Texture`,  
`Win360::D3D9IndexBuffer`, `Win360::D3D9VertexBuffer`, and  
`Models::Model`.

`bool`  
`Resources::Resource::Save()` [virtual, inherited]

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.
int ( ) const [inline, inherited]
Core::RefCounted::GetRefCount

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Base::TextureBase::MapInfo
Base::TextureBase::MapInfo Class Reference

#include <texturebase.h>
Detailed Description

access info filled by Map methods
Public Member Functions

**MapInfo ()**

*constructor*
Base::TransformDeviceBase
Base::TransformDeviceBase Class Reference

#include <transformdevicebase.h>

Inheritance diagram for Base::TransformDeviceBase:
Detailed Description

Manages global transform matrices and their combinations. Input transforms are the view transform (transforms from world to view space), the projection transform (describes the projection from view space into projection space (pre-div-z)) and the current model matrix (transforms from model to world space). From these input transforms, the TransformDevice computes all useful combinations and inverted versions.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformDeviceBase ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~TransformDeviceBase ()</td>
<td>destructor</td>
</tr>
<tr>
<td>bool Open ()</td>
<td>open the transform device</td>
</tr>
<tr>
<td>void Close ()</td>
<td>close the transform device</td>
</tr>
<tr>
<td>bool IsOpen () const</td>
<td>return true if device is open</td>
</tr>
<tr>
<td>void ApplyViewSettings ()</td>
<td>apply view dependent settings</td>
</tr>
<tr>
<td>void ApplyModelTransforms (const Ptr&lt;</td>
<td>apply any model transform needed, implementation is platform dependend</td>
</tr>
<tr>
<td>CoreGraphics::ShaderInstance &gt; &amp;shdInst)</td>
<td></td>
</tr>
<tr>
<td>void SetProjTransform (const Math::matrix44 &amp;m)</td>
<td>set projection transform</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; GetProjTransform ()</td>
<td>get current projection matrix</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; GetInvProjTransform ()</td>
<td>get inverted projection transform</td>
</tr>
<tr>
<td>void SetViewTransform (const Math::matrix44 &amp;m)</td>
<td>set view transform</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; GetViewTransform ()</td>
<td>get view transform</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; GetInvViewTransform ()</td>
<td>get current inverted view transform</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; GetViewProjTransform ()</td>
<td>get current view-projection transform</td>
</tr>
<tr>
<td>void SetModelTransform (const Math::matrix44 &amp;m)</td>
<td>set model transform</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; GetModelTransform ()</td>
<td></td>
</tr>
</tbody>
</table>
get current model transform

const Math::matrix44 & GetInvModelTransform ()
get current inverted model transform

const Math::matrix44 & GetModelViewTransform ()
get current model-view matrix

const Math::matrix44 & GetInvModelViewTransform ()
get current inverted model-view-transform

const Math::matrix44 & GetModelViewProjTransform ()
get current model-view-projection transform

void SetFocalLength (const Math::float2 &len)
set focal length

const Math::float2 & GetFocalLength () const
get focal length

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name
<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
get the current refcount
Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
get the class name
Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Base::VertexBufferBase
#include <vertexbufferbase.h>

Inheritance diagram for Base::VertexBufferBase:
Detailed Description

A resource which holds an array of vertices.

(C) 2006 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Usage</th>
<th>resource usage flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>State</td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>VertexBufferBase()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~VertexBufferBase()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <code>Unload()</code></td>
<td>Unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td>void <code>Map(MapType mapType)</code></td>
<td>Map the vertices for CPU access</td>
</tr>
<tr>
<td>void <code>Unmap()</code></td>
<td>Unmap the resource</td>
</tr>
<tr>
<td>void <code>SetVertexLayout(const CoreGraphics::VertexLayout &amp;vertexLayout)</code></td>
<td>Set vertex layout (set by resource loader)</td>
</tr>
<tr>
<td>const <code>CoreGraphics::VertexLayout</code> &amp; <code>GetVertexLayout()</code></td>
<td>Get the vertex layout</td>
</tr>
<tr>
<td>void <code>SetNumVertices(SizeT numVertices)</code></td>
<td>Set number of vertices (set by resource loader)</td>
</tr>
<tr>
<td><code>SizeT GetNumVertices()</code></td>
<td>Get number of vertices in the buffer</td>
</tr>
<tr>
<td>void <code>SetUsage(Usage usage)</code></td>
<td>Set resource usage type</td>
</tr>
<tr>
<td><code>Usage GetUsage()</code></td>
<td>Get resource usage type</td>
</tr>
<tr>
<td>void <code>SetAccess(Access access)</code></td>
<td>Set resource CPU access type</td>
</tr>
<tr>
<td><code>Access GetAccess()</code></td>
<td>Get CPU access type</td>
</tr>
<tr>
<td>void <code>SetAsyncEnabled(bool b)</code></td>
<td>Request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsAsyncEnabled()</code></td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>void Lock()</code></td>
<td>set locked to true</td>
</tr>
<tr>
<td><code>void Unlock()</code></td>
<td>set locked to false</td>
</tr>
<tr>
<td><code>bool IsLocked()</code></td>
<td>returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td><code>void SetResourceId(const ResourceId &amp;id)</code></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><code>const ResourceId &amp; GetResourceId()</code></td>
<td>get the resource identifier</td>
</tr>
<tr>
<td><code>void SetLoader(const Ptr&lt;ResourceLoader&gt; &amp;loader)</code></td>
<td>set optional resource loader</td>
</tr>
<tr>
<td><code>const Ptr&lt;ResourceLoader&gt; &amp; GetLoader()</code></td>
<td>get optional resource loader</td>
</tr>
<tr>
<td><code>void SetSaver(const Ptr&lt;ResourceSaver&gt; &amp;saver)</code></td>
<td>set optional resource saver</td>
</tr>
<tr>
<td><code>const Ptr&lt;ResourceSaver&gt; &amp; GetSaver()</code></td>
<td>get optional resource saver</td>
</tr>
<tr>
<td><code>SizeT GetUseCount()</code></td>
<td>get current use count</td>
</tr>
<tr>
<td><code>virtual State Load()</code></td>
<td>load the resource</td>
</tr>
<tr>
<td><code>void SetState(State s)</code></td>
<td>set current state (usually only called during Load())</td>
</tr>
<tr>
<td><code>State GetState()</code></td>
<td>get current state</td>
</tr>
<tr>
<td><code>bool IsLoaded()</code></td>
<td>return true if current state is Loaded</td>
</tr>
<tr>
<td><code>bool IsPending()</code></td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td>C++ Type</td>
<td>Method Name</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>bool</td>
<td>LoadFailed ()</td>
</tr>
<tr>
<td>virtual bool</td>
<td>Save ()</td>
</tr>
<tr>
<td>int</td>
<td>GetRefCount ()</td>
</tr>
<tr>
<td>void</td>
<td>AddRef ()</td>
</tr>
<tr>
<td>void</td>
<td>Release ()</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Rtti &amp;rtti)</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;className)</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC)</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti)</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName)</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC)</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName ()</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC ()</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
# Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><strong>IncrUseCount</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>increment use count</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>DecrUseCount</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>decrement use count</em></td>
</tr>
</tbody>
</table>
void * Base::VertexBufferBase::Map ( MapType mapType )

map the vertices for CPU access

Make the vertex buffer content accessible by the CPU. The vertex buffer must have been initialized with the right Access and Usage flags (see parent class for details). There are several reasons why a mapping the resource may fail, this depends on the platform (for instance, the resource may currently be busy, or selected for rendering).

Reimplemented in Win360::D3D9VertexBuffer.

void Base::VertexBufferBase::Unmap ( )

unmap the resource

Give up CPU access on the vertex buffer content.

Reimplemented in Win360::D3D9VertexBuffer.

bool Resources::Resource::Save ( ) [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the IsPending() method will return true as long as the load is in progress, and IsLoaded() will become true when the loading process has finished. If the load has failed, IsPending() will switch to false and IsLoaded() will not be true.
save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Base::VertexLayoutBase
#include <vertexlayoutbase.h>

Inheritance diagram for Base::VertexLayoutBase:
Detailed Description

**Base** class for platform-specific vertex component subclasses. This allows subclasses to add platform-specific information to vertex components.

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>VertexLayoutBase ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~VertexLayoutBase ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void Setup(const Util::Array&lt;CoreGraphics::VertexComponent&gt; setup)</code></td>
<td>setup the vertex layout</td>
</tr>
<tr>
<td><code>bool IsValid () const</code></td>
<td>return true if valid has been setup</td>
</tr>
<tr>
<td><code>SizeT GetNumComponents () const</code></td>
<td>get number of components</td>
</tr>
<tr>
<td><code>const CoreGraphics::VertexComponent &amp; GetComponentAt (IndexT i)</code></td>
<td>get vertex component at index</td>
</tr>
<tr>
<td><code>bool HasComponent (CoreGraphics::VertexComponent::SemanticName semName, IndexT semIndex)</code></td>
<td>return true if vertex component exists</td>
</tr>
<tr>
<td><code>IndexT FindComponent (CoreGraphics::VertexComponent::SemanticName semName, IndexT semIndex)</code></td>
<td>get index of vertex component by semantics</td>
</tr>
<tr>
<td><code>SizeT GetVertexByteSize () const</code></td>
<td>get the vertex stride in number of bytes</td>
</tr>
<tr>
<td><code>const Util::Array&lt;CoreGraphics::VertexComponent&gt; &amp; GetVertexComponents ()</code></td>
<td>get vertex components</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp; rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;str) const</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;fourcc) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks ()</strong></th>
</tr>
</thead>
</table>

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Protected Member Functions

void **Discard** ()

*discard the vertex layout object*
**Static Protected Member Functions**

<table>
<thead>
<tr>
<th>static Util::String</th>
<th><strong>BuildSignature</strong> (const Util::Array/CoreGraphics::VertexComponent &gt; &amp;c)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get sharing signature for a set of vertex components</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
get the class name
Get the class name of the object.
```

```cpp
const Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
```
builds only!)

This method should be called as the very last before an application exits.
#include <vertexlayoutserverbase.h>

Inheritance diagram for Base::VertexLayoutServerBase:
Detailed Description

The **VertexLayoutServerBase** creates VertexLayout objects shared by their vertex component signature. On some platforms it is more efficient to share VertexLayout objects across meshes with identical vertex structure. Note that there is no way to manually discard vertex components. Vertex components stay alive for the life time of the application until the **Close()** method of the **VertexLayoutServerBase** is called.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VertexLayoutServerBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~VertexLayoutServerBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Open ()</strong></td>
<td>open the server</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>close the server</td>
</tr>
<tr>
<td><strong>IsOpen ()</strong> const</td>
<td>return true if open</td>
</tr>
<tr>
<td><strong>CreateSharedVertexLayout</strong> (const <strong>Ptr&lt; CoreGraphics::VertexLayout &gt;</strong>) &amp;vertexComponents)</td>
<td>create shared vertex layout object</td>
</tr>
<tr>
<td><strong>GetRefCount ()</strong> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>Util::String</strong> &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**BaseGameFeature::AnimEventManager**
BaseGameFeature::AnimEventManager
Class Reference

#include <animeventmanager.h>

Inheritance diagram for BaseGameFeature::AnimEventManager:
Detailed Description

This application side manager, polls every frame all anim events.

(C) 2009 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AnimEventManager()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>virtual ~AnimEventManager()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>Util::Array&lt; Animation::AnimEventInfo &gt;</code></td>
<td>get all anim events for entity</td>
</tr>
<tr>
<td><code>GetAnimEventsByCategory(const Ptr&lt;Game::Entity&gt;&amp; entity, const Util::StringAtom&amp; category)</code></td>
<td>get all anim events for entity</td>
</tr>
<tr>
<td><code>virtual void OnActivate()</code></td>
<td>called when attached to game server</td>
</tr>
<tr>
<td><code>virtual void OnDeactivate()</code></td>
<td>called when removed from game server</td>
</tr>
<tr>
<td><code>bool IsActive()</code></td>
<td>return true if currently active</td>
</tr>
<tr>
<td><code>virtual void OnBeginFrame()</code></td>
<td>called before frame by the game server</td>
</tr>
<tr>
<td><code>virtual void OnFrame()</code></td>
<td>called per-frame by the game server</td>
</tr>
<tr>
<td><code>virtual void OnEndFrame()</code></td>
<td>called after frame by the game server</td>
</tr>
<tr>
<td><code>virtual void OnLoad()</code></td>
<td>called after loading game state</td>
</tr>
<tr>
<td><code>virtual void OnSave()</code></td>
<td>called before saving game state</td>
</tr>
<tr>
<td><code>virtual void OnStart()</code></td>
<td>called by Game::Server::Start() when the world is started</td>
</tr>
<tr>
<td><code>virtual void OnRenderDebug()</code></td>
<td>render a debug visualization</td>
</tr>
<tr>
<td><code>virtual void HandleMessage(const Ptr&lt;Messaging::Message&gt;&amp; msg)</code></td>
<td>handle a single message (distribute to ports which accept the message)</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void AttachPort (const Ptr&lt; Port &gt;&amp;port)</code></td>
<td>Attach a message port</td>
</tr>
<tr>
<td><code>void RemovePort (const Ptr&lt; Port &gt;&amp;port)</code></td>
<td>Remove a message port</td>
</tr>
<tr>
<td><code>bool HasPort (const Ptr&lt; Port &gt;&amp;port)</code> const</td>
<td>Return true if a port exists</td>
</tr>
<tr>
<td><code>virtual void SetupAcceptedMessages ()</code></td>
<td>Override to register accepted messages</td>
</tr>
<tr>
<td><code>void AttachHandler (const Ptr&lt; Handler &gt;&amp;h)</code></td>
<td>Attach a message handler to the port</td>
</tr>
<tr>
<td><code>void RemoveHandler (const Ptr&lt; Handler &gt;&amp;h)</code></td>
<td>Remove a message handler from the port</td>
</tr>
<tr>
<td><code>void RemoveAllHandlers ()</code></td>
<td>Remove all message handlers from the port</td>
</tr>
<tr>
<td><code>SizeT GetNumHandlers () const</code></td>
<td>Return number of handlers attached to the port</td>
</tr>
<tr>
<td><code>const Ptr&lt; Handler &gt;&amp; GetHandlerAtIndex (IndexT i)</code> const</td>
<td>Get a message handler by index</td>
</tr>
<tr>
<td><code>virtual void Send (const Ptr&lt; Message &gt;&amp;msg)</code></td>
<td>Send a message to the port</td>
</tr>
<tr>
<td><code>const Util::Array&lt; const Id * &gt;&amp; GetAcceptedMessages () const</code></td>
<td>Get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td><code>bool AcceptsMessage (const Id &amp;msgId)</code> const</td>
<td>Return true if port accepts this message</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Release ()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

`void RegisterMessage (const Id &msgId)`

*register a single accepted message*
Member Function Documentation

```cpp
void Game::Manager::OnActivate( ) [virtual, inherited]
called when attached to game server

This method is called when the manager is attached to the game server. The manager base class will register its message port with the message server.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager, BaseGameFeature::EnvQueryManager, BaseGameFeature::GlobalAttrsManager, Script::DialogManager, and Script::ScriptManager.

void Game::Manager::OnDeactivate( ) [virtual, inherited]
called when removed from game server

This method is called when the manager is removed from the game server. It will unregister its message port from the message server at this point.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager, BaseGameFeature::EntityManager, BaseGameFeature::EnvEntityManager, BaseGameFeature::EnvQueryManager, Script::DialogManager, and Script::ScriptManager.

void Game::Manager::OnBeginFrame( ) [virtual, inherited]
called before frame by the game server

Called before frame, override in subclasses
Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Game::Manager::OnEndFrame() [virtual, inherited]
```
called after frame by the game server

Called after frame, override in subclasses

Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Messaging::Dispatcher::HandleMessage(const Messaging::Message &msg) [virtual, inherited]
```
handle a single message (distribute to ports which accept the message)

Handle a message. The message will only be distributed to ports which accept the message.

Reimplemented from **Messaging::Port**.

Reimplemented in **Script::DialogManager**.

```cpp
void Messaging::Dispatcher::AttachPort(const Port &port) [inherited]
```
attach a message port

Attach a new message port.

**Parameters:**

- `port` pointer to a message port object

```cpp
void Messaging::Dispatcher::RemovePort(const Port &port) [inherited]
```
remove a message port
Remove a message port object.

**Parameters:**

`handler` pointer to message port object to be removed

```cpp
bool Messaging::Dispatcher::HasPort(const Ptr<Port> &port) const [inherited]
```

return true if a port exists

Return true if a port is already attached.

```cpp
void Messaging::Port::AttachHandler(const Ptr<Handler> &h) [inherited]
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler(const Ptr<Handler> &h) [inherited]
```

remove a message handler from the port

Remove a message handler from the port.

```cpp
void Messaging::Port::Send(const Ptr<Message> &msg) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**BaseGameFeature::CategoryManager**
BaseGameFeature::CategoryManager
Class Reference

#include <categorymanager.h>

Inheritance diagram for BaseGameFeature::CategoryManager:
Detailed Description

Wraps entity categories and provides access to category template and instance tables.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CategoryManager ()</strong></td>
<td>virtual constructor</td>
</tr>
<tr>
<td><strong>~CategoryManager ()</strong></td>
<td>virtual destructor</td>
</tr>
<tr>
<td><strong>OnActivate ()</strong></td>
<td>called when attached to game server</td>
</tr>
<tr>
<td><strong>OnDeactivate ()</strong></td>
<td>called when removed from game server</td>
</tr>
<tr>
<td><strong>CommitChangesToDatabase ()</strong></td>
<td>commit changes back into database</td>
</tr>
<tr>
<td><strong>LoadInstances (const Util::String &amp;levelName)</strong></td>
<td>load all instances with the given level attribute</td>
</tr>
<tr>
<td><strong>FindInstances (const Util::String &amp;levelName)</strong></td>
<td>find all instances with the given level attribute</td>
</tr>
<tr>
<td><strong>CreateDummyInstance ()</strong></td>
<td>create a dummy instance which will never be saved to the database</td>
</tr>
<tr>
<td><strong>CreateInstanceFromTemplate (const Util::String &amp;categoryName, const Util::String &amp;id)</strong></td>
<td>create a new instance from a template</td>
</tr>
<tr>
<td><strong>CreateInstanceFromTemplateAsCategory</strong></td>
<td>(const Util::String &amp;categoryName, const Util::String &amp;id, const Util::String &amp;targetCategory)</td>
</tr>
<tr>
<td><strong>CreateInstanceFromAttrs</strong></td>
<td>(const Util::String &amp;categoryName, const Util::Array<a href="">Attr::Attribute</a> &amp;attrs)</td>
</tr>
<tr>
<td>Entry</td>
<td>CreateInstanceFromInstance (const Entry &amp;source)</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>create a new instance as a copy from another instance of the same category</td>
</tr>
<tr>
<td>Entry</td>
<td>CreateInstanceFromInstanceAsCategory (const Entry &amp;source, const Util::String &amp;targetCategory, bool createMissingAttributes=false)</td>
</tr>
<tr>
<td></td>
<td>create a new instance as a copy from another instance, but in a different category</td>
</tr>
<tr>
<td>Util::Array&lt;Entry&gt;</td>
<td>GetInstancesByAttr (const Attr::Attribute &amp;attrs, bool loadedOnly, bool firstMatchOnly)</td>
</tr>
<tr>
<td></td>
<td>find and/or create new instance by matching a key attribute, this method can be quite slow!</td>
</tr>
<tr>
<td>Util::Array&lt;Entry&gt;</td>
<td>GetInstancesByAttrs (const Util::Array<a href="">Attr::Attribute</a> &amp;attrs, bool loadedOnly, bool firstMatchOnly)</td>
</tr>
<tr>
<td></td>
<td>find and/or create new instance by matching a key attribute, this method can be quite slow!</td>
</tr>
<tr>
<td>void</td>
<td>DeleteInstance (const Entry &amp;source)</td>
</tr>
<tr>
<td></td>
<td>delete an instance identified by GUID</td>
</tr>
<tr>
<td>int</td>
<td>GetNumInstances () const</td>
</tr>
<tr>
<td></td>
<td>get current overall number of instances</td>
</tr>
<tr>
<td>void</td>
<td>SetInstanceEntity (const Entry &amp;instance, const Ptr<a href="">Game::Entity</a> &amp;entity)</td>
</tr>
<tr>
<td></td>
<td>bind entity pointer to instance</td>
</tr>
<tr>
<td>Ptr<a href="">Game::Entity</a></td>
<td>GetInstanceEntity (const Entry &amp;instance) const</td>
</tr>
<tr>
<td></td>
<td>get instance entity pointer (can be 0)</td>
</tr>
<tr>
<td>bool</td>
<td>HasCategory (const Util::String &amp;categoryName) const</td>
</tr>
<tr>
<td></td>
<td>return true if a category exists</td>
</tr>
<tr>
<td>SizeT</td>
<td>GetNumCategories () const</td>
</tr>
<tr>
<td></td>
<td>get number of categories</td>
</tr>
<tr>
<td>const Category &amp;</td>
<td>GetCategoryByIndex (IndexT index) const</td>
</tr>
<tr>
<td></td>
<td>access to category attributes by index</td>
</tr>
<tr>
<td>const Category &amp;</td>
<td>GetCategoryByName (const Util::String</td>
</tr>
<tr>
<td><strong>Function</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><code>void BeginAddCategoryAttrs(const Util::String &amp;categoryName)</code></td>
<td>Begin adding category attributes</td>
</tr>
<tr>
<td><code>void AddCategoryAttr(const Attr::AttrId &amp;attrId)</code></td>
<td>Add a category attribute</td>
</tr>
<tr>
<td><code>void EndAddCategoryAttrs()</code></td>
<td>End adding category attributes</td>
</tr>
<tr>
<td><code>bool HasTemplateTable(const Util::String &amp;categoryName)</code></td>
<td>Return true if a template table for a category exists</td>
</tr>
<tr>
<td><code>const Ptr&lt;Db::ValueTable&gt; &amp;GetTemplateTable(const Util::String &amp;categoryName)</code></td>
<td>Get template value table by category name</td>
</tr>
<tr>
<td><code>Entry FindTemplate(const Util::String &amp;categoryName, const Util::String &amp;id)</code></td>
<td>Find a template entry by category name and id</td>
</tr>
<tr>
<td><code>Entry FindTemplateByAttr(const Attr::Attribute &amp;attr, const Util::String &amp;categoryName=&quot;&quot;)</code></td>
<td>Find a single template by attribute</td>
</tr>
<tr>
<td><code>bool HasInstanceTable(const Util::String &amp;categoryName)</code></td>
<td>Return true if an instance table for a category exists</td>
</tr>
<tr>
<td><code>constPtr&lt;Db::ValueTable&gt; &amp;GetInstanceTable(const Util::String &amp;categoryName)</code></td>
<td>Get instance value table by category name</td>
</tr>
<tr>
<td><code>void DuplicateLevel(const Util::String &amp;levelName, const Util::String &amp;duplicateName)</code></td>
<td>Duplicates a complete level in the databases</td>
</tr>
<tr>
<td><code>void DeleteLevel(const Util::String &amp;levelName)</code></td>
<td>Deletes a complete level from database</td>
</tr>
<tr>
<td><code>RenameLevel(const Util::String &amp;levelName)</code></td>
<td>Rename level</td>
</tr>
</tbody>
</table>
void &levelName, const Util::String &newLevelName)  
renames a complete level in the database

bool LevelExists (const Util::String &levelName)  
return true if level is found in _Instance_Levels table

bool IsActive () const  
return true if currently active

virtual void OnBeginFrame ()  
called before frame by the game server

virtual void OnFrame ()  
called per-frame by the game server

virtual void OnEndFrame ()  
called after frame by the game server

virtual void OnLoad ()  
called after loading game state

virtual void OnSave ()  
called before saving game state

virtual void OnStart ()  
called by Game::Server::Start() when the world is started

virtual void OnRenderDebug ()  
render a debug visualization

virtual void HandleMessage (const Ptr< Messaging::Message > &msg)  
handle a single message (distribute to ports which accept the message)

void AttachPort (const Ptr< Port > &port)  
attach a message port

void RemovePort (const Ptr< Port > &port)  
remove a message port

bool HasPort (const Ptr< Port > &port) const  
return true if a port exists

virtual void SetupAcceptedMessages ()  
override to register accepted messages

void AttachHandler (const Ptr< Handler > &h)  
attach a message handler to the port

void RemoveHandler (constPtr< Handler > &h)
remove a message handler from the port

void **RemoveAllHandlers** ()
remove all message handler from the port

SizeT **GetNumHandlers** () const
return number of handlers attached to the port

const **GetHandlerAtIndex** (IndexT i) const
get a message handler by index

virtual void **Send** (const **Message** &msg)
send a message to the port

const Util::Array< const Id * > & **GetAcceptedMessages** () const
get the array of accepted messages (sorted)

bool **AcceptsMessage** (const Id &msgId) const
return true if port accepts this msg

int **GetRefCount** () const
get the current refcount

void **AddRef** ()
increment refcount by one

void **Release** ()
decrement refcount and destroy object if refcount is zero

bool **IsInstanceOf** (const Rtti &rtti) const
return true if this object is instance of given class

bool **IsInstanceOf** (const Util::String &className) const
return true if this object is instance of given class, or a derived class

bool **IsA** (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool **IsA** (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool **IsA** (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const **GetClassName** () const
get the class name
<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
</table>

*get the class FourCC code*
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounitng leaks, call at end of application (NEBULA3_DEBUG builds only!)
Protected Member Functions

void RegisterMessage (const Id &msgId)

register a single accepted message
# Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>Category</th>
<th>describes a category [More...]</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>Entry</td>
<td>describes an entry in a value table [More...]</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void BaseGameFeature::CategoryManager::OnActivate() [virtual]
called when attached to game server

NOTE: The category manager must be re-activated when the world database is opened so that it may read the categories from the world database.

Reimplemented from Game::Manager.
```

```cpp
void BaseGameFeature::CategoryManager::OnDeactivate() [virtual]
called when removed from game server

This unloads everything.

Reimplemented from Game::Manager.
```

```cpp
void BaseGameFeature::CategoryManager::CommitChangesToDatabase()
commit changes back into database

This commits changes to the instance data back into the database. Call this method right before creating a save game.
```

```cpp
CategoryManager::Entry
BaseGameFeature::CategoryManager::CreateDummyInstance()
create a dummy instance which will never be saved to the database

Create an instance in the dummy category, this is for entities which are created as helper entities and should never show up in the database (i.e. the global environment entity).
```

```cpp
CategoryManager::Entry
```
create a new instance from a template

This creates a new instance from a category name and id and returns its location.

create a new instance from a template as a different category

This creates a new instance from a category name and id, but puts the instance into a different category's instance table.

create a new instance as a copy from another instance of the same category

This creates a new instance as a copy of another instance of the same category.
create a new instance as a copy from another instance, but in a different category

This creates a new instance in a different category as a copy of another instance.

```cpp
void BaseGameFeature::CategoryManager::DeleteInstance (const Entry entry&)
```
delete an instance identified by GUID

This deletes an instance from the database.

```cpp
int BaseGameFeature::CategoryManager::GetNumInstances (const)
```
get current overall number of instances

This returns the overall number of loaded instances.

```cpp
void BaseGameFeature::CategoryManager::SetInstanceEntity (const Entry instance, const Ptr<Game::Entity> entity &)
```
bind entity pointer to instance

Associate a game entity pointer with an instance. This uses the ValueTable's per-row user data field to store the pointer.

```cpp
Ptr<Game::Entity> BaseGameFeature::CategoryManager::GetInstanceEntity (const Entry instance &)
```
get instance entity pointer (can be 0!)

Get the game entity pointer associated with an instance.
begin adding category attributes

Begin adding category attributes.

add a category attribute

This adds an attribute to a category. Will extend the template and instance table of the category by the new attribute and update the attr/category mapping table.

end adding category attributes

End adding attributes to a category.

find a single template by attribute

Find the first template which has the matching attribute. If categoryName is given, only search in this category, otherwise in all categories which have the attribute.

WARNING: this method does linear searches on the value tables and thus can be slow.
Game::Manager::OnBeginFrame( ) [virtual, inherited]
called before frame by the game server
Called before frame, override in subclasses
Reimplemented in BaseGameFeature::EntityManager.

void Game::Manager::OnEndFrame( ) [virtual, inherited]
called after frame by the game server
Called after frame, override in subclasses
Reimplemented in BaseGameFeature::EntityManager.

void Messaging::Dispatcher::HandleMessage(const Ptr<Messaging::Message> &msg ) [virtual, inherited]
handle a single message (distribute to ports which accept the message)
Handle a message. The message will only be distributed to ports which accept the message.
Reimplemented from Messaging::Port.
Reimplemented in Script::DialogManager.

void Messaging::Dispatcher::AttachPort(const Ptr<Port> &port ) [inherited]
attach a message port
Attach a new message port.

Parameters:

   port  pointer to a message port object
void Messaging::Dispatcher::RemovePort (const Ptr<Port>& port) [inherited]

remove a message port

Remove a message port object.

**Parameters:**

`handler` pointer to message port object to be removed

bool Messaging::Dispatcher::HasPort (const Ptr<Port>& port) const [inherited]

return true if a port exists

Return true if a port is already attached.

void Messaging::Port::AttachHandler (const Ptr<Handler>& h) [inherited]

attach a message handler to the port

Attach a message handler to the port.

void Messaging::Port::RemoveHandler (const Ptr<Handler>& h) [inherited]

remove a message handler from the port

Remove a message handler from the port.
send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::CategoryManager::Category
BaseGameFeature::CategoryManager::Category

Class Reference

#include <categorymanager.h>
Detailed Description

describes a category
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetName () const</code></td>
<td>get category name</td>
</tr>
<tr>
<td><code>bool IsVirtual () const</code></td>
<td>return true if virtual category (starting with a . in the db.xml)</td>
</tr>
<tr>
<td><code>bool IsSpecial () const</code></td>
<td>return true if this is a special category (e.g. _Environment)</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetTemplateTableName () const</code></td>
<td>get template table name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetInstanceTableName () const</code></td>
<td>get instance table name</td>
</tr>
<tr>
<td><code>bool HasTemplateDataset () const</code></td>
<td>return true if the category has a template dataset</td>
</tr>
<tr>
<td><code>bool HasInstanceDataset () const</code></td>
<td>return true if the category has an instance dataset</td>
</tr>
<tr>
<td><code>const Ptr&lt; Db::Dataset &gt; &amp; GetTemplateDataset () const</code></td>
<td>get pointer to template dataset</td>
</tr>
<tr>
<td><code>const Ptr&lt; Db::Dataset &gt; &amp; GetInstanceDataset () const</code></td>
<td>get pointer to instance dataset</td>
</tr>
</tbody>
</table>
BaseGameFeature::CategoryManager::Entry
BaseGameFeature::CategoryManager::Entry

#include <categorymanager.h>
Detailed Description

describes an entry in a value table
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entry ()</strong></td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><strong>Entry (const <strong>Util::String</strong> &amp;categoryName,</strong> const <strong>Ptr</strong>&lt; Db::ValueTable &gt; &amp;valueTable,** IndexT rowIndex)**</td>
<td><em>constructor with value table and row index</em></td>
</tr>
<tr>
<td>bool <strong>IsValid ()</strong> const</td>
<td><em>return true if valid</em></td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>Category ()</strong> const</td>
<td><em>get category name</em></td>
</tr>
<tr>
<td>const <strong>Ptr</strong>&lt; Db::ValueTable &gt; &amp; <strong>Values ()</strong> const</td>
<td><em>get pointer to value table</em></td>
</tr>
<tr>
<td>IndexT <strong>RowIndex ()</strong> const</td>
<td><em>get row index in value table</em></td>
</tr>
</tbody>
</table>
BaseGameFeature::EntityLoader
BaseGameFeature::EntityLoader Class Reference

#include <entityloader.h>

Inheritance diagram for BaseGameFeature::EntityLoader:

```
  Core::RefCounted

  BaseGameFeature::EntityLoaderBase

  BaseGameFeature::EntityLoader
```
Detailed Description

**Loader** helper for universal game entities. The properties which are attached to the entity are described in blueprints.xml, the attributes to attach come from the world database.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool Load (const Util::Array&lt; Util::String &gt; &amp;activeLayers)</td>
<td>load entity objects into the level</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static bool</td>
<td><strong>IsLoading()</strong></td>
<td><em>is loader currently inside Load Function</em></td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks()</strong></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th><code>EntityIsInActiveLayer</code> (const <code>Ptr&lt;Db::ValueTable&gt;</code> &amp;values, IndexT rowIndex, const <code>Util::Array&lt;Util::String&gt;</code> &amp;levelActiveLayers) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if entity at current <code>Db::Reader pos</code> is in active layer</td>
</tr>
</tbody>
</table>

```cpp
bool EntityIsInActiveLayer (const Ptr<Db::ValueTable> &values, IndexT rowIndex, const Util::Array<Util::String> &levelActiveLayers) const

return true if entity at current Db::Reader pos is in active layer
```
Member Function Documentation

```cpp
bool BaseGameFeature::EntityLoaderBase::EntityIsInActiveLayer( const Ptr<Db::ValueTable> values, IndexT rowIndex, const Util::Array<Util::String> & levelActiveLayers ) const [protected, inherited]
```

return true if entity at current Db::Reader pos is in active layer

Return true if the entity at the current Db::Reader position is in the active layer set.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::EntityLoaderBase
BaseGameFeature::EntityLoaderBase
Class Reference

#include <entityloaderbase.h>

Inheritance diagram for BaseGameFeature::EntityLoaderBase:

- Core::RefCounted
  - BaseGameFeature::EntityLoaderBase
    - BaseGameFeature::EntityLoader
    - BaseGameFeature::EnvironmentLoader
Detailed Description

Abstract loader helper for game entities.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EntityLoaderBase ()</code></td>
<td></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~EntityLoaderBase ()</code></td>
<td></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Load (const Util::Array&lt; Util::String &gt; &amp;activeLayers)</code></td>
<td></td>
<td>Load entity objects into the level</td>
</tr>
<tr>
<td><code>GetRefCount ()</code> const</td>
<td></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>AddRef ()</code></td>
<td></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release ()</code></td>
<td></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Rtti &amp;rtti)</code> const</td>
<td></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::String &amp;className)</code> const</td>
<td></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::FourCC &amp;classFourCC)</code> const</td>
<td></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti)</code> const</td>
<td></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA (const Util::String &amp;rttiName)</code> const</td>
<td></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td><code>Util::String &amp;</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td><code>Util::FourCC</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static bool</td>
<td>IsLoading ()</td>
<td><em>is loader currently inside Load Function</em></td>
</tr>
<tr>
<td>static void</td>
<td>DumpRefCountingLeaks ()</td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>EntityIsInActiveLayer</strong> (const <strong>Ptr</strong>&lt; Db::ValueTable &gt; &amp;values, IndexT rowIndex, const <strong>Util::Array</strong>&lt; <strong>Util::String</strong> &gt; &amp;levelActiveLayers) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if entity at current Db::Reader pos is in active layer</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool BaseGameFeature::EntityLoaderBase::EntityIsInActiveLayer
    (const Ptr<Db::ValueTable> & values, const IndexT rowIndex, const Util::Array<Util::String> & levelActiveLayers)
return true if entity at current Db::Reader pos is in active layer
Return true if the entity at the current Db::Reader position is in the active layer set.
```

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::EntityManager
BaseGameFeature::EntityManager
Class Reference

#include <entitymanager.h>

Inheritance diagram for BaseGameFeature::EntityManager:
Detailed Description

The entity manager object keeps track of all active game entities and calls their per-frame-update methods to keep them alive. It also contains methods to iterate through existing entities. Derive from this class if your application needs different or more advanced game entity management, but make sure that all methods which are defined in entity manager still do the expected thing in your derived class.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EntityManager ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>virtual ~EntityManager ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void SetMaxTriggerDistance (float d)</code></td>
<td>TEMP HACK: set the maximum-trigger-distance (default is 100 meters).</td>
</tr>
<tr>
<td><code>float GetMaxTriggerDistance () const</code></td>
<td>TEMP HACK: get the maximum-trigger-distance.</td>
</tr>
<tr>
<td><code>virtual void OnDeactivate ()</code></td>
<td>called when removed from game server</td>
</tr>
<tr>
<td><code>void OnStart ()</code></td>
<td>called on start by game server</td>
</tr>
<tr>
<td><code>virtual void OnBeginFrame ()</code></td>
<td>called per-frame by game server</td>
</tr>
<tr>
<td><code>virtual void OnEndFrame ()</code></td>
<td>called per-frame by game server</td>
</tr>
<tr>
<td><code>virtual void OnLoad ()</code></td>
<td>called after level loaded</td>
</tr>
<tr>
<td><code>virtual void OnSave ()</code></td>
<td>called before saving game state</td>
</tr>
<tr>
<td><code>void OnRenderDebug ()</code></td>
<td>called if a render debug visualization is requested</td>
</tr>
<tr>
<td><code>void Cleanup ()</code></td>
<td>cleanup the entity manager, removes/deactivates all entities</td>
</tr>
<tr>
<td><code>bool HasActiveEntities () const</code></td>
<td>return true if at least one active entity exists in the world</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Ptr&lt;Game::Entity&gt; &gt; &amp; GetEntities () const</code></td>
<td>get currently active entities</td>
</tr>
<tr>
<td><code>void AttachEntity (const Ptr&lt;Game::Entity&gt; &amp;entity)</code></td>
<td></td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RemoveEntity (const Ptr<a href="">Game::Entity</a> &amp;entity)</td>
<td>remove an entity from the world, delayed until end of frame</td>
</tr>
<tr>
<td>DeleteEntity (const Ptr<a href="">Game::Entity</a> &amp;entity)</td>
<td>delete an entity from the world (also deletes the entity from DB!), delayed until end of frame</td>
</tr>
<tr>
<td>RemoveEntityImmediate (const Ptr<a href="">Game::Entity</a> &amp;entity)</td>
<td>remove an entity from the world, calling restrictions apply</td>
</tr>
<tr>
<td>DeleteEntityImmediate (const Ptr<a href="">Game::Entity</a> &amp;entity)</td>
<td>delete an entity from the world (also deletes the entity from DB!), calling restrictions apply</td>
</tr>
<tr>
<td>ExistsEntityByUniqueId (Game::Entity::EntityId id) const</td>
<td>return true if a entity with the given unique id exists</td>
</tr>
<tr>
<td>GetEntityByUniqueId (Game::Entity::EntityId id) const</td>
<td>get the entity for the given unique id</td>
</tr>
<tr>
<td>ExistsEntityByAttr (const Attr::Attribute &amp;attr) const</td>
<td>return true if at least one entity exists with the given attribute</td>
</tr>
<tr>
<td>GetEntitiesByAttr (const Attr::Attribute &amp;attr, bool onlyFirstEntity=false)</td>
<td>get the entities for the given attribute</td>
</tr>
<tr>
<td>GetEntitiesByAttrs (const Util::Array<a href="">Attr::Attribute</a> &amp;attr, bool onlyFirstEntity=false)</td>
<td>get a single entity by multiple matching attributes</td>
</tr>
<tr>
<td>GetEntityByAttr (const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>Ptr&lt;Game::Entity&gt;</code></td>
<td>Attr::Attribute &amp;attr, bool failOnError=true) get a single entity by a single attribute</td>
</tr>
<tr>
<td><code>GetEntityByAttrs (const Util::Array&lt;Attr::Attribute&gt; &amp;attr, bool failOnError=true)</code></td>
<td>get a single entity by multiple matching attributes</td>
</tr>
<tr>
<td><code>GetEntitiesInActivityBubble (Util::Array&lt;Ptr&lt;Game::Entity&gt;&gt; &amp;outEntities)</code></td>
<td>fill provided array with all entities inside the activity bubble</td>
</tr>
<tr>
<td><code>IsEntityInDelayedJobs (const Ptr&lt;Game::Entity&gt; &amp;entity)</code></td>
<td>returns true if entity is in delayed jobs for delete or remove</td>
</tr>
<tr>
<td><code>virtual void OnActivate ()</code></td>
<td>called when attached to game server</td>
</tr>
<tr>
<td><code>bool IsActive () const</code></td>
<td>return true if currently active</td>
</tr>
<tr>
<td><code>virtual void OnFrame ()</code></td>
<td>called per-frame by the game server</td>
</tr>
<tr>
<td><code>virtual void HandleMessage (const Ptr&lt;Messaging::Message&gt; &amp;msg)</code></td>
<td>handle a single message (distribute to ports which accept the message)</td>
</tr>
<tr>
<td><code>void AttachPort (const Ptr&lt;Port&gt; &amp;port)</code></td>
<td>attach a message port</td>
</tr>
<tr>
<td><code>void RemovePort (const Ptr&lt;Port&gt; &amp;port)</code></td>
<td>remove a message port</td>
</tr>
<tr>
<td><code>bool HasPort (const Ptr&lt;Port&gt; &amp;port) const</code></td>
<td>return true if a port exists</td>
</tr>
<tr>
<td><code>virtual void SetupAcceptedMessages ()</code></td>
<td>override to register accepted messages</td>
</tr>
</tbody>
</table>
void AttachHandler (const Ptr<
Handler> &h)
attach a message handler to the port

void RemoveHandler (const Ptr<
Handler> &h)
remove a message handler from the port

void RemoveAllHandlers ()
remove all message handler from the port

SizeT GetNumHandlers () const
return number of handlers attached to the port

const Ptr< Handler> &
GetHandlerAtIndex (IndexT i) const
get a message handler by index

virtual void Send (const Ptr< Message> &msg)
send a message to the port

const Util::Array< const Id *> &
GetAcceptedMessages () const
get the array of accepted messages (sorted)

bool AcceptsMessage (const Id &msgid) const
return true if port accepts this msg

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const
<table>
<thead>
<tr>
<th>bool</th>
<th><strong>Util::FourCC</strong> &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>Util::String</strong> &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
### Protected Types

<table>
<thead>
<tr>
<th>enum</th>
<th>DelayedJobType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>delayed job types</em></td>
</tr>
</tbody>
</table>
**Protected Member Functions**

- `void StartEntities ()`
  - call `OnStart()` on all entities

- `void ActivateEntity (const Ptr<Game::Entity> &entity)`
  - activate an entity

- `void DeactivateEntity (const_PTR<Game::Entity> &entity)`
  - deactivate an entity

- `void AddDelayedJob (DelayedJobType type, const_PTR<Game::Entity> &entity)`
  - add a delayed job (Remove or Delete entity)

- `void HandleDelayedJobs ()`
  - handle all queued up delayed jobs, called at beginning and end of frame

- `void RemoveNullEntriesFromArrays ()`
  - remove all null entries from the internal arrays

- `void UpdateTriggeredEntities ()`
  - update both arrays

- `bool IsInFocus (const_PTR/Game::Entity> &entity, Math::point &focusEntityPos)`
  - check if entity is near focus entity

- `void RemoveEntityFromTriggered (const_PTR/Game::Entity> &entity)`
  - remove entity from triggerd/untriggered arrays

- `void RegisterMessage (const Id &msgId)`
  - register a single accepted message
<table>
<thead>
<tr>
<th>class</th>
<th>DelayedJob</th>
</tr>
</thead>
</table>

*delayed jobs*

More...
Member Function Documentation

```cpp
void BaseGameFeature::EntityManager::OnBeginFrame() [virtual]

called per-frame by game server

Handles delayed jobs, like removing entities. Calls OnBeginFrame and OnMoveBefore on all Entities. OnMoveAfter and OnRender is called in OnEndFrame of the `EntityManager` to allow any physics system to step its world.

Reimplemented from `Game::Manager`.

```cpp
void BaseGameFeature::EntityManager::OnEndFrame() [virtual]

called per-frame by game server

Calls OnMoveAfter and OnRender on all Entities.

Reimplemented from `Game::Manager`.

```cpp
void BaseGameFeature::EntityManager::Cleanup()

cleanup the entity manager, removes/deactivates all entities

This immediately removes and deactivates all entities from the entity manager. This method is usually only called at the end of a level.

```cpp
bool BaseGameFeature::EntityManager::HasActiveEntities() const

return true if at least one active entity exists in the world

Return true if there is at least one active entity in the world.

```cpp
const Util::Array< Ptr< Game::Entity > >& BaseGameFeature::EntityManager::GetEntities() const
```
get currently active entities

Get the array of currently active entities.

```cpp
void BaseGameFeature::EntityManager::AttachEntity
    (const Ptr<Game::Entity> &entity)
```

Immediately attach an entity to the world

Immediately attach (and activate) a game entity.

```cpp
void BaseGameFeature::EntityManager::RemoveEntity
    (const Ptr<Game::Entity> &entity)
```

remove an entity from the world, delayed until end of frame

Remove a game entity from the entity manager. This will just mark the entity as dismissed, deactivation will happen at the beginning of the next frame to prevent any data inconsistencies. The entity will just function as usual for the rest of the frame.

```cpp
void BaseGameFeature::EntityManager::DeleteEntity
    (const Ptr<Game::Entity> &entity)
```

delete an entity from the world (also deletes the entity from DB!), delayed until end of frame

Delete an entity (remove it from the world and delete it from the database) at the beginning of the next frame.

```cpp
void BaseGameFeature::EntityManager::RemoveEntityImmediate
    (const Ptr<Game::Entity> &entity)
```

remove an entity from the world, calling restrictions apply

Immediately remove an entity from the entity manager. This method is more restrictive than `RemoveEntity()` because it must not be called from inside the game loop.
void BaseGameFeature::EntityManager::DeleteEntityImmediate (const Ptr<Game::Entity> &entity)

delete an entity from the world (also deletes the entity from DB!), calling restrictions apply

Immediately delete an entity (remove from world and delete from database). This method is more restrictive than DeleteEntity() because it must not be called from inside the game loop.

Ptr<Game::Entity> BaseGameFeature::EntityManager::GetEntityByUniqueId (Game::Entity::EntityId uniqueld)

going the entity for the given unique id

Return entity by unique id or an invalid Ptr<> if the entity doesn't exist.

Util::Array<Ptr<Game::Entity> > BaseGameFeature::EntityManager::GetEntitiesInActivityBubble (Game::Entity outEntities)

fill provided array with all entities inside the activity bubble

This updates updates the activityBubbleEntities array which contains all entities around the current viewer which should be triggered.

25-Jan-07 floh added optional per-entity trigger radius

void BaseGameFeature::EntityManager::StartEntities ( ) [protected]

call OnStart() on all entities

Invoke the OnStart() method on all active entities.

void BaseGameFeature::EntityManager::DeactivateEntity (const Ptr<Game::Entity> &entity) [protected]

deactivate an entity
Deactivate an entity. This will remove the entity from the world and call its `OnDeactivate()` method. This is a private helper method and will be called from `RemoveEntity()` or `RemoveDismissedEntities()`.

```cpp
void BaseGameFeature::EntityManager::AddDelayedJob (DelayedJobType type,
const Ptr<Game::Entity>& entity
) [protected]
```

add a delayed job (Remove or Delete entity)

Add a new delayed job to the internal job queue. Delayed jobs are executed at the beginning of the next frame.

```cpp
void BaseGameFeature::EntityManager::HandleDelayedJobs ( ) [protected]
```

handle all queued up delayed jobs, called at beginning and end of frame

This handles all delayed jobs. This method is executed once at the end of each frame.

```cpp
void BaseGameFeature::EntityManager::RemoveNullEntriesFromArray ( ) [protected]
```

remove all null entries from the internal arrays

This method should be called once at the end of the frame. It will check the internal arrays for null entries, and remove those entries. Null entries are written by the `RemoveEntityImmediate()` method because it may be dangerous to change array layouts while an iteration is running over the array.

```cpp
void BaseGameFeature::EntityManager::UpdateTriggeredEntities ( ) [protected]
```

update both arrays

This updates updates the triggeredEntities/untriggeredEntities arrays
that divide all entities around the current viewer in those who should be triggered and those who shouldn't be triggered.

```cpp
bool BaseGameFeature::EntityManager::IsInFocus(
    const Ptr<Game::Entity> curEntity,
    & Math::point focusEntityPos
) [protected]
```

check if entity is near focus entity

Checks if an entity is "in reach" of the focus entity, i.e. within Attr::EntityTriggerRadius + this->maxTriggerDistance.

Note: will also return true on entities without transform attribute

```cpp
void BaseGameFeature::EntityManager::RemoveEntityFromTriggered(
    const Ptr<Game::Entity> entity
) [protected]
```

remove entity from triggerd/untriggered arrays

Immediately remove an entity from the entity manager. This method is more restrictive than `RemoveEntity()` because it must not be called from inside the game loop.

```cpp
void Game::Manager::OnActivate() [virtual, inherited]
```

called when attached to game server

This method is called when the manager is attached to the game server. The manager base class will register its message port with the message server.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager, BaseGameFeature::EnvQueryManager, BaseGameFeature::GlobalAttrsManager, Script::DialogManager, and Script::ScriptManager.
void Messaging::Dispatcher::HandleMessage (const Ptr<Messaging::Message> &msg) [virtual, inherited]

handle a single message (distribute to ports which accept the message)

Handle a message. The message will only be distributed to ports which accept the message.

Reimplemented from Messaging::Port.

Reimplemented in Script::DialogManager.

void Messaging::Dispatcher::AttachPort (const Ptr<Port> &port) [inherited]

attach a message port

Attach a new message port.

**Parameters:**

* port pointer to a message port object

void Messaging::Dispatcher::RemovePort (const Ptr<Port> &port) [inherited]

remove a message port

Remove a message port object.

**Parameters:**

* handler pointer to message port object to be removed

bool Messaging::Dispatcher::HasPort (const Ptr<Port> &port) const [inherited]
return true if a port exists

Return true if a port is already attached.

```cpp
void Messaging::Port::AttachHandler ( const Ptr<Handler> & h ) [inherited]
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler ( const Ptr<Handler> & h ) [inherited]
```

remove a message handler from the port

Remove a message handler from the port.

```cpp
void Messaging::Port::Send ( const Ptr<Message> & msg ) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the 
HandleMessage() method of all attached handlers. If the message 
has been handled by at least one of the handlers, the Handled() flag 
of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
const Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

BaseGameFeature::EntityManager::DelayedJob
BaseGameFeature::EntityManager::Delay

Class Reference

#include <entitymanager.h>
Detailed Description

delayed jobs
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DelayedJob ()</code></td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><code>DelayedJob (DelayedJobType t, const Ptr&lt;Game::Entity&gt; &amp;e)</code></td>
<td><em>constructor</em></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](https://www.doxygen.org) at Fri Mar 26 15:21:41 2010
BaseGameFeature::EnvEntityManager
BaseGameFeature::EnvEntityManager
Class Reference

#include <enventitymanager.h>

Inheritance diagram for BaseGameFeature::EnvEntityManager:
Detailed Description

Manages creation and updating of environment entities. All simple environment entities are kept in a single game entity to prevent pollution of the game entity pool with entities that don't actually do anything. Non-simple environment entities (entities with animations or physics) will still be created as normal game entities.

The EnvEntityManager basically hides all differences between those types of environment entities.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EnvEntityManager()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~EnvEntityManager()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <code>OnDeactivate()</code></td>
<td>called when removed from game server</td>
</tr>
<tr>
<td><code>CreateEnvEntity(const Util::Array&lt;Attr::Attribute&gt; &amp;attrs)</code></td>
<td>create environment entity from scratch from provided attributes</td>
</tr>
<tr>
<td><code>CreateEnvEntity(const Ptr&lt;Db::ValueTable&gt; &amp;instTable, IndexT instTableRowIndex)</code></td>
<td>create an environment entity and attach to world</td>
</tr>
<tr>
<td>bool <code>EnvEntityExists(const Util::String &amp;id)</code> const</td>
<td>check if an environment entity exists by _ID</td>
</tr>
<tr>
<td>void <code>SetEnvEntityTransform(const Util::String &amp;id, const Math::matrix44 &amp;m)</code></td>
<td>set the world transform of an environment entity</td>
</tr>
<tr>
<td>void <code>DeleteEnvEntity(const Util::String &amp;id)</code></td>
<td>delete environment with given id</td>
</tr>
<tr>
<td><code>Util::Array&lt;Ptr&lt;Graphics::ModelEntity&gt;&gt;</code></td>
<td>get graphics entities by id (may return empty array)</td>
</tr>
<tr>
<td><code>OnActivate()</code></td>
<td>clear environment entity</td>
</tr>
<tr>
<td>virtual void <code>OnActivate()</code></td>
<td></td>
</tr>
</tbody>
</table>
bool IsActive () const
return true if currently active

virtual void OnBeginFrame ()
called before frame by the game server

virtual void OnFrame ()
called per-frame by the game server

virtual void OnEndFrame ()
called after frame by the game server

virtual void OnLoad ()
called after loading game state

virtual void OnSave ()
called before saving game state

virtual void OnStart ()
called by Game::Server::Start() when the world is started

virtual void OnRenderDebug ()
render a debug visualization

virtual void HandleMessage (const Ptr<Messaging::Message> &msg)
handle a single message (distribute to ports which accept the message)

void AttachPort (const Ptr<Port> &port)
attach a message port

void RemovePort (const Ptr<Port> &port)
remove a message port

bool HasPort (const Ptr<Port> &port) const
return true if a port exists

virtual void SetupAcceptedMessages ()
override to register accepted messages

void AttachHandler (const Ptr<Handler> &h)
attach a message handler to the port

void RemoveHandler (const Ptr<Handler> &h)
remove a message handler from the port
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Handler &gt; &amp;h)</code></td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td><code>void RemoveAllHandlers ()</code></td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td><code>SizeT GetNumHandlers () const</code></td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td><code>const Ptr&lt; Handler &gt; &amp; GetHandlerAtIndex (IndexT i)</code></td>
<td>get a message handler by index</td>
</tr>
<tr>
<td><code>virtual void Send (const Ptr&lt; Message &gt; &amp;msg)</code></td>
<td>send a message to the port</td>
</tr>
<tr>
<td><code>const Util::Array&lt; const Id * &gt; &amp; GetAcceptedMessages () const</code></td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td><code>bool AcceptsMessage (const Id &amp;msgId)</code></td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by FourCC</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool <code>IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC <code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Protected Member Functions

```c
void RegisterMessage (const Id &msgId)
```

*register a single accepted message*
Member Function Documentation

```cpp
void BaseGameFeature::EnvEntityManager::CreateEnvEntity(
    const Util::Array<Attr::Attribute> &attrs)
```

create environment entity from scratch from provided attributes

Create an simple environment entity from scratch. The following attributes should be provided:

- Attr::Graphics
- Attr::Physics
- Attr::ID
- Attr::Transform
- Attr::AnimPath

```cpp
void BaseGameFeature::EnvEntityManager::CreateEnvEntity(
    const Ptr<Db::ValueTable> &instTable,
    IndexT instTableRowIndex)
```

create an environment entity and attach to world

Create an environment entity from its database attributes. This could end up in a bunch of graphics entities and collide shapes pooled in the central env entity, or as a traditional game object (if the environment object is animated or has physics).

```cpp
bool BaseGameFeature::EnvEntityManager::EnvEntityExists(
    const Util::String &id) const
```

check if an environment entity exists by _ID

Return true if an environment entity with the given id exists.

```cpp
void BaseGameFeature::EnvEntityManager::SetEnvEntityTransform(
    const Util::String &id)
```
const 
Math::matrix44 m
&
)

set the world transform of an environment entity

Set transformation of an environment entity.

void 
BaseGameFeature::EnvEntityManager::DeleteEnvEntity (const 
Util::String id &)

delete environment with given id

Delete an environment entity.

Util::Array< Ptr< Graphics::ModelEntity >>
BaseGameFeature::EnvEntityManager::GetGraphicsEntities (const 
Util::String id &)

get graphics entities by id (may return empty array)

Returns the graphics entities of the given environment entity.

void 
Game::Manager::OnActivate ( ) [virtual, inherited]

called when attached to game server

This method is called when the manager is attached to the game server. The manager base class will register its message port with the message server.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager, BaseGameFeature::EnvQueryManager, BaseGameFeature::GlobalAttrsManager, Script::DialogManager, and Script::ScriptManager.

void 
Game::Manager::OnBeginFrame ( ) [virtual, inherited]
called before frame by the game server

Called before frame, override in subclasses

Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Game::Manager::OnEndFrame() [virtual, inherited]
```

called after frame by the game server

Called after frame, override in subclasses

Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Messaging::Dispatcher::HandleMessage(const Messaging::Message &msg) [virtual, inherited]
```

handle a single message (distribute to ports which accept the message)

Handle a message. The message will only be distributed to ports which accept the message.

Reimplemented from **Messaging::Port**.

Reimplemented in **Script::DialogManager**.

```cpp
void Messaging::Dispatcher::AttachPort(const Ptr<Port> &port) [inherited]
```

attach a message port

Attach a new message port.

**Parameters:**

- `port` pointer to a message port object
Messaging::Dispatcher::RemovePort

remove a message port

Remove a message port object.

**Parameters:**

- **handler** pointer to message port object to be removed

bool
Messaging::Dispatcher::HasPort(const Ptr<Port>&)

return true if a port exists

Return true if a port is already attached.

void
Messaging::Port::AttachHandler(const Ptr<Handler>&)

attach a message handler to the port

Attach a message handler to the port.

void
Messaging::Port::RemoveHandler(const Ptr<Handler>&)

remove a message handler from the port

Remove a message handler from the port.

void
Messaging::Port::Send(const Ptr<Message>&)

send a message to the port
Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC code of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::EnvironmentLoader
BaseGameFeature::EnvironmentLoader
Class Reference

#include <environmentloader.h>

Inheritance diagram for BaseGameFeature::EnvironmentLoader:
Detailed Description

Helper class which loads all the environment objects into a level. Called by `BaseGameFeature::LevelLoader`.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool</td>
<td>Load (const Util::Array&lt; Util::String &gt; &amp;activeLayers)</td>
<td>Load environment objects into the level</td>
</tr>
<tr>
<td>int</td>
<td>GetRefCount () const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td>AddRef ()</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void</td>
<td>Release ()</td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;className) const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
<td>Get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static bool</td>
<td>IsLoading()</td>
<td><code>is loader currently inside Load Function</code></td>
</tr>
<tr>
<td>static void</td>
<td>DumpRefCountingLeaks()</td>
<td><code>dump refcounting leaks, call at end of application (NEBUA3_DEBUG builds only!)</code></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>EntityIsInActiveLayer</strong> (const <strong>Ptr</strong>&lt; Db::ValueTable &gt; &amp;values, const IndexT rowIndex, const <strong>Util::Array</strong>&lt; <strong>Util::String</strong> &gt; &amp;levelActiveLayers) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if entity at current Db::Reader pos is in active layer</td>
</tr>
</tbody>
</table>

---
Member Function Documentation

```cpp
bool BaseGameFeature::EntityLoaderBase::EntityIsActiveInLayer(const Ptr<Db::ValueTable> & values, IndexT rowIndex, const Util::Array<Util::String> & levelActiveLayers)
```

return true if entity at current Db::Reader pos is in active layer

Return true if the entity at the current Db::Reader position is in the active layer set.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::EnvQueryManager
BaseGameFeature::EnvQueryManager
Class Reference

#include <envquerymanager.h>

Inheritance diagram for BaseGameFeature::EnvQueryManager:
Detailed Description

The **EnvQueryManager** implements environment queries into the game world, like stabbing queries, line-of-sight checks, etc...

(C) 2005 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong><code>EnvQueryManager()</code></strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual <code>~EnvQueryManager()</code></strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual void <code>OnActivate()</code></strong></td>
<td>called when attached to game server</td>
</tr>
<tr>
<td><strong>virtual void <code>OnDeactivate()</code></strong></td>
<td>called when removed from game server</td>
</tr>
<tr>
<td><strong>virtual void <code>OnFrame()</code></strong></td>
<td>called per-frame by game server</td>
</tr>
<tr>
<td><strong><code>Game::Entity * GetEntityUnderMouse()</code></strong></td>
<td>get the entity under the mouse cursor</td>
</tr>
<tr>
<td><strong>const <code>Math::point &amp; GetMousePos3d()</code></strong></td>
<td>get the mouse position in the 3d world</td>
</tr>
<tr>
<td><strong>Math::line <code>ComputeMouseWorldRay</code></strong></td>
<td>(const <code>Math::float2 &amp;mousePos</code>, float length, const <code>Ptr&lt; Graphics::View &gt; &amp;view</code>)</td>
</tr>
<tr>
<td><strong>const <code>Math::vector &amp; GetUpVector()</code></strong></td>
<td>get the upVector of the face under the mouse cursor</td>
</tr>
<tr>
<td><strong>bool <code>HasMouselntersection()</code></strong></td>
<td>return true if mouse is over &quot;something&quot;</td>
</tr>
<tr>
<td><strong>Util::Array&lt; Ptr&lt; Game::Entity &gt; &gt; GetEntitiesInSphere</strong></td>
<td>(const <code>Math::point &amp;midPoint</code>, float radius)</td>
</tr>
<tr>
<td><strong>Util::Array&lt; Ptr&lt; Game::Entity &gt; &gt; GetEntitiesInBox</strong></td>
<td>(const <code>Math::vector &amp;scale</code>, const <code>Math::matrix44 &amp;m</code>)</td>
</tr>
<tr>
<td><strong>bool IsActive()`</strong></td>
<td>return true if currently active</td>
</tr>
<tr>
<td><strong>virtual void <code>OnBeginFrame()</code></strong></td>
<td>called before frame by the game server</td>
</tr>
</tbody>
</table>
virtual void OnEndFrame ()
called after frame by the game server

virtual void OnLoad ()
called after loading game state

virtual void OnSave ()
called before saving game state

virtual void OnStart ()
called by Game::Server::Start() when the world is started

virtual void OnRenderDebug ()
render a debug visualization

virtual void HandleMessage (const Ptr<Message::Message> &msg)
handle a single message (distribute to ports which accept the message)

void AttachPort (const Ptr<Port> &port)
attach a message port

void RemovePort (const Ptr<Port> &port)
remove a message port

bool HasPort (const Ptr<Port> &port) const
return true if a port exists

virtual void SetupAcceptedMessages ()
override to register accepted messages

void AttachHandler (const Ptr<Handler> &h)
attach a message handler to the port

void RemoveHandler (const Ptr<Handler> &h)
remove a message handler from the port

void RemoveAllHandlers ()
remove all message handler from the port

SizeT GetNumHandlers () const
return number of handlers attached to the port

const Ptr<Handler> & GetHandlerAtIndex (IndexT i) const
get a message handler by index

virtual void Send (const Ptr<Message> &msg)
send a message to the port
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const Util::Array&lt; const Id * &gt; &amp; GetAcceptedMessages()</code> const</td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td><code>bool AcceptsMessage (const Id &amp;msgId)</code> const</td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td><code>int GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName()</code> const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC()</code> const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

void RegisterMessage (const Id &msgId)

register a single accepted message
Member Function Documentation

```cpp
void BaseGameFeature::EnvQueryManager::OnFrame() [virtual]

called per-frame by game server

This method is called per-frame by the game server and updates the current values, like entity-under-mouse, 3d mouse position, etc...

Reimplemented from `Game::Manager`.

```cpp
Entity * BaseGameFeature::EnvQueryManager::GetEntityUnderMouse() const

get the entity under the mouse cursor

This returns a pointer to the entity under the mouse, or 0 if mouse is not over an entity.

```cpp
const point & BaseGameFeature::EnvQueryManager::GetMousePos3d() const

get the mouse position in the 3d world

This returns the position where a vector through the mouse position intersects the 3d world (or the nearest entity). If the mouse doesn't intersect, the result will be undefined, and the method `HasMouseIntersection()` returns false.

```cpp
const vector & BaseGameFeature::EnvQueryManager::GetUpVector() const

get the upVector of the face under the mouse cursor

This returns the upvector of the face under the mousecursor. If the mouse doesn't intersect, the result will be undefined, and the method `HasMouseIntersection()` returns false.

bool
BaseGameFeature::EnvQueryManager::HasMouseIntersection(const) returns true if mouse is over "something"

Returns true if the vector through the current mouse position intersects the world, or an entity, false if no intersection exists.

```cpp
Util::Array< Ptr< Game::Entity >> BaseGameFeature::EnvQueryManager::GetEntitiesInSphere(const Math::point midPoint, float radius)
```

get all entities in a given spherical area

Returns all game entities which intersect the given sphere. Uses the physics subsystem to do the query.

```cpp
Util::Array< Ptr< Game::Entity >> BaseGameFeature::EnvQueryManager::GetEntitiesInBox(const Math::vector scale, const Math::matrix44 m)
```

get all entities in a given box shaped area

Returns all game entities which intersect the given box. Uses the physics subsystem to do the query.

```cpp
void Game::Manager::OnBeginFrame()
```
called before frame by the game server

Called before frame, override in subclasses

Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Game::Manager::OnEndFrame()
```
called after frame by the game server

Called after frame, override in subclasses

Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Messaging::Dispatcher::HandleMessage(const Messaging::Message msg) [virtual, inherited]
```

handle a single message (distribute to ports which accept the message)

Handle a message. The message will only be distributed to ports which accept the message.

Reimplemented from **Messaging::Port**.

Reimplemented in **Script::DialogManager**.

```cpp
void Messaging::Dispatcher::AttachPort(const Port &port) [inherited]
```

attach a message port

Attach a new message port.

**Parameters:**

- **port** pointer to a message port object

```cpp
void Messaging::Dispatcher::RemovePort(const Port &port) [inherited]
```

remove a message port

Remove a message port object.

**Parameters:**

- **handler** pointer to message port object to be removed
bool Messaging::Dispatcher::HasPort ( const Ptr<Port> & port ) const [inherited]

return true if a port exists

Return true if a port is already attached.

void Messaging::Port::AttachHandler ( const Ptr<Handler> & h ) [inherited]

attach a message handler to the port

Attach a message handler to the port.

void Messaging::Port::RemoveHandler ( const Ptr<Handler> & h ) [inherited]

remove a message handler from the port

Remove a message handler from the port.

void Messaging::Port::Send ( const Ptr<Message> & msg ) [virtual, inherited]

send a message to the port

Send a message to the port. This will immediately call the HandleMessage() method of all attached handlers. If the message has been handled by at least one of the handlers, the Handled() flag of the message will be set to true.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount
Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::FactoryManager
#include <factorymanager.h>

Inheritance diagram for BaseGameFeature::FactoryManager:

```
BaseGameFeature::FactoryManager
  ^
  |  
  v  
Game::Manager
  ^
  |  
  v  
Messaging::Dispatcher
  ^
  |  
  v  
Messaging::Port
  ^
  |  
  v  
Core::RefCounted
```
Detailed Description

The FactoryManager is responsible for creating new game entities. FactoryManager must be extended by Mangalore applications if the application needs new game entity classes.

The FactoryManager loads the file

data:tables/blueprints.xml

on creation, which contains the construction blueprints for the entity types of your application. This file defines entity types by the properties which are added to them.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FactoryManager ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~FactoryManager ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt;Game::Entity&gt;</code> <strong>CreateEntityByClassName</strong> (const <code>Util::String &amp;cppClassName</code>) const</td>
<td>create a new raw game entity by type name, extend this method in subclasses for new types</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt;Game::Entity&gt;</code> <strong>CreateEntityByCategory</strong> (const <code>Util::String &amp;categoryName</code>, const <code>Ptr&lt;Db::ValueTable&gt; &amp;attrTable</code>, <code>IndexT attrTableRowIndex</code>, <code>bool onlyDistributedProperties=false</code>) const</td>
<td>create a new entity from its category name</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt;Game::Entity&gt;</code> <strong>CreateEntityByAttrs</strong> (const <code>Util::String &amp;categoryName</code>, const <code>Util::Array&lt;Attr::Attribute&gt; &amp;attrs</code>, <code>bool onlyDistributedProperties=false</code>) const</td>
<td>create a new entity from scratch and initialize it with the provided attributes</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt;Game::Entity&gt;</code> <strong>CreateEntityByTemplate</strong> (const <code>Util::String &amp;categoryName</code>, const <code>Util::String &amp;templateName</code>, <code>bool onlyDistributedProperties=false</code>) const</td>
<td>create a new entity from a database template entry</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt;Game::Entity&gt;</code> <strong>CreateEntityByTemplateAsCategory</strong> (const <code>Util::String &amp;categoryName</code>, const <code>Util::String &amp;templateName</code>, const <code>Util::String &amp;targetCategory</code>) const</td>
<td>create a new entity from a database template entry, and add it into a different category</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt;Game::Entity&gt;</code> <strong>CreateEntityByEntity</strong> (const <code>Ptr&lt;Game::Entity&gt; &amp;sourceEntity</code>) const</td>
<td>create a new entity cloning an existing one</td>
</tr>
</tbody>
</table>
| virtual `Ptr<Game::Entity>` **CreateEntityByEntityAsCategory** (const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual <code>Ptr&lt;Game::Entity&gt;</code> &amp;sourceEntity, const Util::String &amp;targetCategory, bool createMissingAttributes=false</td>
<td>create a new entity in a different category cloning an existing one</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt;Game::Entity&gt;</code></td>
<td><code>CreateEntityByKeyAttr</code> (const Attr::Attribute &amp;key) const</td>
</tr>
<tr>
<td></td>
<td>create new entity from world database using any key attribute</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt;Game::Entity&gt;</code></td>
<td><code>CreateEntityByGuid</code> (const Util::Guid &amp;guid) const</td>
</tr>
<tr>
<td></td>
<td>create new entity from world database using GUID as key attribute</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt;Game::Property&gt;</code></td>
<td><code>CreateProperty</code> (const Util::String &amp;type) const</td>
</tr>
<tr>
<td></td>
<td>create a new property by type name, extend in subclass for new types</td>
</tr>
<tr>
<td>void</td>
<td><code>AddProperties</code> (const <code>Ptr&lt;Game::Entity&gt;</code> &amp;entity, const Util::String &amp;categoryName, bool onlyRemoteProperties) const</td>
</tr>
<tr>
<td></td>
<td>add properties to entity according to blue print</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>SetupAttributes</strong> ()</td>
</tr>
<tr>
<td></td>
<td>setup attributes on properties</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnActivate</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called when attached to game server</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnDeactivate</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called when removed from game server</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsActive</strong> ()</td>
</tr>
<tr>
<td></td>
<td>return true if currently active</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnBeginFrame</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called before frame by the game server</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnFrame</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called per-frame by the game server</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnEndFrame</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called after frame by the game server</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnLoad</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called after loading game state</td>
</tr>
</tbody>
</table>
virtual void OnSave ()
called before saving game state

virtual void OnStart ()
called by Game::Server::Start() when the world is started

virtual void OnRenderDebug ()
render a debug visualization

virtual void HandleMessage (const Ptr<Message::Message> &msg)
handle a single message (distribute to ports which accept the message)

void AttachPort (const Ptr<Port> &port)
attach a message port

void RemovePort (const Ptr<Port> &port)
remove a message port

bool HasPort (const Ptr<Port> &port) const
return true if a port exists

virtual void SetupAcceptedMessages ()
override to register accepted messages

void AttachHandler (const Ptr<Handler> &h)
attach a message handler to the port

void RemoveHandler (const Ptr<Handler> &h)
remove a message handler from the port

void RemoveAllHandlers ()
remove all message handler from the port

SizeT GetNumHandlers () const
return number of handlers attached to the port

const Ptr<Handler> & GetHandlerAtIndex (IndexT i) const
get a message handler by index

virtual void Send (const Ptr<Message> &msg)
send a message to the port

const Util::Array<const Id*> & GetAcceptedMessages () const
get the array of accepted messages (sorted)

bool AcceptsMessage (const Id &msgId) const
return true if port accepts this msg

int GetRefCount () const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddRef ()</strong></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA (const Rtti &amp;rtti) const</strong></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA (const Util::String &amp;rttiName) const</strong></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><strong>GetClassName () const</strong></td>
<td>Get the class name</td>
</tr>
<tr>
<td><strong>GetClassFourCC () const</strong></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetBlueprintsFilename</code></td>
<td>static void SetBlueprintsFilename (const Util::String &amp;name) set a optional blueprints.xml, which is used instead of standard blueprint.xml</td>
</tr>
<tr>
<td><code>DumpRefCountingLeaks</code></td>
<td>static void DumpRefCountingLeaks () dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th><code>ParseBluePrints()</code></th>
<th>parse entity blueprints file</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexT</td>
<td><code>FindBluePrint(const Util::String &amp;entityType)</code> const</td>
<td>find blueprint index by property type</td>
</tr>
<tr>
<td>void</td>
<td><code>RegisterMessage(const Id &amp;msgId)</code></td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Data Structures

struct PropertyEntry

an entity blueprint, these are created by ParseBlueprints() More...
Member Function Documentation

\textbf{Ptr< Entity >}  
\texttt{BaseGameFeature::FactoryManager::CreateEntityByClassName} (const \texttt{Util::String} \texttt{cppClassName} &)

create a new raw game entity by type name, extend this method in subclasses for new types

Create a raw entity by its C++ class name name. This method should be extended by subclasses if a Mangalore application implements new \texttt{Game::Entity} subclasses.

\textbf{Ptr< Entity >}  
\texttt{BaseGameFeature::FactoryManager::CreateEntityByCategory} (const \texttt{Util::String} & \texttt{categoryName}, const \texttt{Ptr< Db::ValueTable>} \texttt{attrTable}, IndexT \texttt{attrTableRowIndex} bool \texttt{onlyRemoteProperties} = false)

create a new entity from its category name

Create an entity from its category name. The category name is looked up in the blueprints.xml file to check what properties must be attached to the entity. All required properties will be attached, and all attributes will be initialised according to the attribute table.

\textbf{Ptr< Entity >}  
\texttt{BaseGameFeature::FactoryManager::CreateEntityByAttrs} (const \texttt{Util::String} & \texttt{categoryName}, const \texttt{Util::Array< Attr::Attribute>} \texttt{attrs}, bool \texttt{onlyRemoteProperties} = false)

\texttt{c}
create a new entity from scratch and initialize it with the provided attributes

Create a new entity from scratch and initialize it with the provided attributes.

```cpp
Ptr< Entity >
BaseGameFeature::FactoryManager::CreateEntityByTemplate
    ( 
        const Util::String categoryName,
        const Util::String templateName,
        bool onlyRemoteProperties = false
    )
```

create a new entity from a database template entry

Create an entity from a template in the database. The template is defined by category name and the template name defined by the the Attr::Id attribute. This will create a complete entity with properties and attributes initialized to the values from the template. A new GUID will be assigned to the entity.

```cpp
Ptr< Entity >
BaseGameFeature::FactoryManager::CreateEntityByTemplateAsCategory
    ( 
        const Util::String categoryName
        const Util::String templateName
        const Util::String targetCat
    )
```

create a new entity from a database template entry, and add it into a different category

Create a new entity from a template, but put the entity into a different category.

```cpp
Ptr< Entity >
const
```
create a new entity cloning an existing one

Create a entity as a clone of a existing one. A new GUID will be assigned.

create a new entity in a different category cloning an existing one

Create a entity as a clone of a existing one but in a different category as the original entity. A new GUID will be assigned.

create new entity from world database using any key attribute

This will 'load' a new entity from the world database. This will create a new entity, attach properties as described by blueprints.xml, and update the entity attributes from the database. Changes to attributes can later be written back to the database by calling the Entity::Save() method.

NOTE: this method will not call the Entity::OnLoad() method, which may be required to finally initialize the entity. The OnLoad() method expects that all other entities in the level have already been loaded, so this must be done after loading in a separate pass.

NOTE: use this method only if you know there's only one matching entity in the database (for instance by Guid attribute), otherwise, use
the CreateEntitiesByKeyAttr() method, which checks all matches.

```cpp
 PTR< Entity >
 BaseGameFeature::FactoryManager::CreateEntityByGuid ( Util::Guid guid ) const [virtual]
```

create new entity from world database using GUID as key attribute

Creates a new entity from the world database using a GUID as key. Simply calls `CreateEntityByKeyAttr()`.

```cpp
 PTR< Property >
 BaseGameFeature::FactoryManager::CreateProperty ( Util::String type ) const [virtual]
```

create a new property by type name, extend in subclass for new types

Create a property by its type name. This method should be extended by subclasses if a Mangalore application implements new properties.

```cpp
 void
 BaseGameFeature::FactoryManager::AddProperties ( const PTR< Game::Entity >& entity,
 const Util::String& categoryName,
 bool onlyRemoteProperties ) const [virtual]
```

add properties to entity according to blue print

This method checks if a blueprint for the provided entity exists, and adds the properties defined in the blue print to the entity. If no matching blueprint exists, the entity will not be altered.

```cpp
 void
 BaseGameFeature::FactoryManager::SetupAttributes ( ) [virtual]
```

setup attributes on properties

Create the properties of every category and call `SetupDefaultAttributes` on it.
bool BaseGameFeature::FactoryManager::ParseBluePrints( ) [protected]

parse entity blueprints file

This method parses the file data:tables/blueprints.xml into the bluePrints array.

IndexT BaseGameFeature::FactoryManager::FindBluePrint( const Util::String entityType & ) const [protected]

find blueprint index by property type

This method finds a blueprint index by entity type name. Returns InvalidIndex if blueprint doesn't exist.

void Game::Manager::OnActivate( ) [virtual, inherited]

called when attached to game server

This method is called when the manager is attached to the game server. The manager base class will register its message port with the message server.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager, BaseGameFeature::EnvQueryManager, BaseGameFeature::GlobalAttrsManager, Script::DialogManager, and Script::ScriptManager.

void Game::Manager::OnDeactivate( ) [virtual, inherited]

called when removed from game server

This method is called when the manager is removed from the game server. It will unregister its message port from the message server at this point.

Reimplemented in BaseGameFeature::TimeManager,
BaseGameFeature::CategoryManager, 
BaseGameFeature::EntityManager, 
BaseGameFeature::EnvEntityManager, 
BaseGameFeature::EnvQueryManager, Script::DialogManager, and Script::ScriptManager.

```cpp
void Game::Manager::OnBeginFrame() [virtual, inherited]
called before frame by the game server

Called before frame, override in subclasses
Reimplemented in BaseGameFeature::EntityManager.
```

```cpp
void Game::Manager::OnEndFrame() [virtual, inherited]
called after frame by the game server

Called after frame, override in subclasses
Reimplemented in BaseGameFeature::EntityManager.
```

```cpp
void Messaging::Dispatcher::HandleMessage(const Ptr<Messaging::Message> &msg) [virtual, inherited]
handle a single message (distribute to ports which accept the message)

Handle a message. The message will only be distributed to ports which accept the message.

Reimplemented from Messaging::Port.

Reimplemented in Script::DialogManager.

```cpp
void Messaging::Dispatcher::AttachPort(const Ptr<Port> &port) [inherited]
```
attach a message port

Attach a new message port.

**Parameters:**

*port* pointer to a message port object

```cpp
void Messaging::Dispatcher::RemovePort(const Ptr<Port>& port) [inherited]
```

remove a message port

Remove a message port object.

**Parameters:**

*handler* pointer to message port object to be removed

```cpp
bool Messaging::Dispatcher::HasPort(const Ptr<Port>& port) const [inherited]
```

return true if a port exists

Return true if a port is already attached.

```cpp
void Messaging::Port::AttachHandler(const Ptr<Handler>& h) [inherited]
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler(const Ptr<Handler>& h) [inherited]
```

remove a message handler from the port
Remove a message handler from the port.

```cpp
void Messaging::Port::Send(const Ptr<Message> & msg ) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC () const [inline, inherited]
```
Core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

BaseGameFeature::FactoryManager::PropertyEntry
BaseGameFeature::FactoryManager::PropertyEntry

#include <factorymanager.h>
Detailed Description

an entity blueprint, these are created by ParseBlueprints()
BaseGameFeature::FocusManager
BaseGameFeature::FocusManager
Class Reference

#include <focusmanager.h>

Inheritance diagram for BaseGameFeature::FocusManager:
Detailed Description

The **FocusManager** singleton object manages the global input and camera focus entities. There may only be one input and camera focus entity at any time, the input focus entity can be different from the camera focus entity.

The input focus entity will be the entity which receives input, the camera focus entity will be the entity which may manipulate the camera.

The **FocusManager** requires an **EntityManager** to iterate through existing entities, and works only on game entities, which have the InputProperty and/or CameraProperty (or a derived class thereof) attached.

Please note that an actual focus switch will happen only once per-frame. This is to avoid chain-reactions when 2 or more objects per frame think they currently have the input focus.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FocusManager ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~FocusManager ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <strong>OnFrame ()</strong></td>
<td>trigger the focus manager once a frame, actual focus switches will happen here</td>
</tr>
<tr>
<td>virtual void <strong>SetFocusEntity</strong> (const <strong>Ptr&lt;</strong> Game::Entity <strong>&gt; &amp;entity)</strong></td>
<td>set input and camera focus to entity, null ptr allowed</td>
</tr>
<tr>
<td>virtual const <strong>Ptr&lt;</strong> Game::Entity <strong>&gt; &amp;</strong> <strong>GetFocusEntity ()</strong> const</td>
<td>get common focus entity, will fail if input != camera focus entity</td>
</tr>
<tr>
<td>virtual void <strong>SetFocusToNextEntity ()</strong></td>
<td>switch input and camera focus to next entity</td>
</tr>
<tr>
<td>virtual void <strong>SetInputFocusEntity</strong> (const <strong>Ptr&lt;</strong> Game::Entity <strong>&gt; &amp;entity)</strong></td>
<td>set the current input focus entity, null ptr allowed</td>
</tr>
<tr>
<td>virtual const <strong>Ptr&lt;</strong> Game::Entity <strong>&gt; &amp;</strong> <strong>GetInputFocusEntity ()</strong> const</td>
<td>get the current input focus entity</td>
</tr>
<tr>
<td>virtual void <strong>SetInputFocusToNextEntity ()</strong></td>
<td>set input focus to next entity</td>
</tr>
<tr>
<td>virtual void <strong>SetCameraFocusEntity</strong> (const <strong>Ptr&lt;</strong> Game::Entity <strong>&gt; &amp;entity)</strong></td>
<td>set the current camera focus entity, null ptr allowed</td>
</tr>
<tr>
<td>virtual const <strong>Ptr&lt;</strong> Game::Entity <strong>&gt; &amp;</strong> <strong>GetCameraFocusEntity ()</strong> const</td>
<td>get the current camera focus entity</td>
</tr>
<tr>
<td>virtual void <strong>SetCameraFocusToNextEntity ()</strong></td>
<td>set camera focus to next entity</td>
</tr>
<tr>
<td>virtual void <strong>OnActivate ()</strong></td>
<td>called when attached to game server</td>
</tr>
<tr>
<td>virtual void <strong>OnDeactivate ()</strong></td>
<td>called when removed from game server</td>
</tr>
<tr>
<td><strong>bool IsActive ()</strong></td>
<td>const</td>
</tr>
</tbody>
</table>
virtual void OnBeginFrame ()
called before frame by the game server

virtual void OnEndFrame ()
called after frame by the game server

virtual void OnLoad ()
called after loading game state

virtual void OnSave ()
called before saving game state

virtual void OnStart ()
called by Game::Server::Start() when the world is started

virtual void OnRenderDebug ()
render a debug visualization

virtual void HandleMessage (const Ptr<Messaging::Message> &msg)
handle a single message (distribute to ports which accept the message)

void AttachPort (const Ptr<Port> &port)
attach a message port

void RemovePort (const Ptr<Port> &port)
remove a message port

bool HasPort (const Ptr<Port> &port) const
return true if a port exists

virtual void SetupAcceptedMessages ()
override to register accepted messages

void AttachHandler (const Ptr<Handler> &h)
attach a message handler to the port

void RemoveHandler (const Ptr<Handler> &h)
remove a message handler from the port

void RemoveAllHandlers ()
remove all message handler from the port

SizeT GetNumHandlers () const
return number of handlers attached to the port

const Ptr<Handler> & GetHandlerAtIndex (IndexT i) const
get a message handler by index

virtual void **Send** (const **Ptr**< **Message** > &msg)
send a message to the port

const **Util::Array**< const **Id** * > & **GetAcceptedMessages** () const
get the array of accepted messages (sorted)

bool **AcceptsMessage** (const **Id** &msgId) const
return true if port accepts this msg

int **GetRefCount** () const
get the current refcount

void **AddRef** ()
increment refcount by one

void **Release** ()
decrement refcount and destroy object if refcount is zero

bool **IsInstanceOf** (const **Rtti** &rtti) const
return true if this object is instance of given class

bool **IsInstanceOf** (const **Util::String** &className) const
return true if this object is instance of given class by string

bool **IsInstanceOf** (const **Util::FourCC** &classFourCC) const
return true if this object is instance of given class by fourcc

bool **IsA** (const **Rtti** &rtti) const
return true if this object is instance of given class, or a derived class

bool **IsA** (const **Util::String** &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool **IsA** (const **Util::FourCC** &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const **Util::String** & **GetClassName** () const
get the class name

**Util::FourCC** **GetClassFourCC** () const
get the class FourCC code
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
**Protected Member Functions**

<table>
<thead>
<tr>
<th>void</th>
<th><code>SwitchFocusEntities()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>actually switch focus entities</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><code>RegisterMessage(const Id &amp;msgId)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void BaseGameFeature::FocusManager::OnFrame() [virtual]
```

trigger the focus manager once a frame, actual focus switches will happen here

This method is called once per frame.

Reimplemented from **Game::Manager**.

```cpp
void BaseGameFeature::FocusManager::SetFocusEntity(const Ptr<Game::Entity> & entity) [virtual]
```

set input and camera focus to entity, null ptr allowed

Sets the input and camera focus to the given entity. The entity pointer may be 0 to clear the input and camera focus. The entity must have both a InputProperty and CameraProperty attached, otherwise the method will fail.

```cpp
const Ptr<Entity> & BaseGameFeature::FocusManager::GetFocusEntity() const [virtual]
```

get common focus entity, will fail if input != camera focus entity

Returns the current common focus entity. This method will fail if the current input focus entity and camera focus entity are not the same. The method may return 0 if there is no current focus entity.

```cpp
void BaseGameFeature::FocusManager::SetFocusToNextEntity() [virtual]
```

switch input and camera focus to next entity

Set focus to next entity which has both an InputProperty and CameraProperty attached. If no current focus entity exists, the method will start to iterate with the first entity. The method will wrap around.
The method will return false if no entities exist which have both an InputProperty and CameraProperty attached.

```cpp
void BaseGameFeature::FocusManager::SetInputFocusEntity(const Ptr<Game::Entity> & entity) [virtual]
```

set the current input focus entity, null ptr allowed

Set input focus entity to the given entity. The entity pointer can be 0, this will clear the current input focus. The entity must have an InputProperty attached for this to work.

```cpp
const Ptr<Entity> & BaseGameFeature::FocusManager::GetInputFocusEntity() const [virtual]
```

get the current input focus entity

Get current input focus entity. This method may return 0 if no input focus entity is set.

```cpp
void BaseGameFeature::FocusManager::SetInputFocusToNextEntity() [virtual]
```

set input focus to next entity

Set input focus to the next entity which has an InputProperty attached.

```cpp
void BaseGameFeature::FocusManager::SetCameraFocusEntity(const Ptr<Game::Entity> & entity) [virtual]
```

set the current camera focus entity, null ptr allowed

Set camera focus entity to the given entity. The entity pointer can be 0, this will clear the current camera focus. The entity must have a CameraProperty attached for this to work.

```cpp
const Ptr<Entity> & BaseGameFeature::FocusManager::GetCameraFocusEntity() const [virtual]
```

get the current camera focus entity
Get current camera focus entity. This method may return 0 if no input focus entity is set.

```cpp
void
BaseGameFeature::FocusManager::SetCameraFocusToNextEntity() [virtual]
```

set camera focus to next entity

Set camera focus to next entity which has a CameraProperty attached.

```cpp
void
BaseGameFeature::FocusManager::SwitchFocusEntities() [protected]
```

actually switch focus entities

Actually switch focus entities. A focus entity switch doesn't happen immediately, but only once per frame. This is to prevent chain-reactions and circular reactions when 2 or more entities think they have the focus in a single frame.

```cpp
void
Game::Manager::OnActivate() [virtual, inherited]
```

called when attached to game server

This method is called when the manager is attached to the game server. The manager base class will register its message port with the message server.

Reimplemented in `BaseGameFeature::TimeManager`, `BaseGameFeature::CategoryManager`, `BaseGameFeature::EnvQueryManager`, `BaseGameFeature::GlobalAttrsManager`, `Script::DialogManager`, and `Script::ScriptManager`.

```cpp
void
Game::Manager::OnDeactivate() [virtual, inherited]
```

called when removed from game server
This method is called when the manager is removed from the game server. It will unregister its message port from the message server at this point.

Reimplemented in **BaseGameFeature::TimeManager**,  
**BaseGameFeature::CategoryManager**,  
**BaseGameFeature::EntityManager**,  
**BaseGameFeature::EnvEntityManager**,  
**BaseGameFeature::EnvQueryManager**, **Script::DialogManager**, and **Script::ScriptManager**.

```cpp
void Game::Manager::OnBeginFrame() [virtual, inherited]
```
called before frame by the game server

Called before frame, override in subclasses

Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Game::Manager::OnEndFrame() [virtual, inherited]
```
called after frame by the game server

Called after frame, override in subclasses

Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Messaging::Dispatcher::HandleMessage(const Ptr<Messaging::Message> msg) [virtual, inherited]
```
handle a single message (distribute to ports which accept the message)

Handle a message. The message will only be distributed to ports which accept the message.

Reimplemented from **Messaging::Port**.

Reimplemented in **Script::DialogManager**.
void Messaging::Dispatcher::AttachPort ( const Ptr<Port> & port ) [inherited]

attach a message port

Attach a new message port.

**Parameters:**

- `port` pointer to a message port object

void Messaging::Dispatcher::RemovePort ( const Ptr<Port> & port ) [inherited]

remove a message port

Remove a message port object.

**Parameters:**

- `handler` pointer to message port object to be removed

bool Messaging::Dispatcher::HasPort ( const Ptr<Port> & port ) const [inherited]

return true if a port exists

Return true if a port is already attached.

void Messaging::Port::AttachHandler ( const Ptr<Handler> & h ) [inherited]

attach a message handler to the port

Attach a message handler to the port.

const
void Messaging::Port::RemoveHandler (Ptr<Handler> & h) [inherited]

remove a message handler from the port

Remove a message handler from the port.

void Messaging::Port::Send (const Ptr<Message> & msg) [virtual, inherited]

send a message to the port

Send a message to the port. This will immediately call the HandleMessage() method of all attached handlers. If the message has been handled by at least one of the handlers, the Handled() flag of the message will be set to true.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
get the class name

Get the class name of the object.

`Util::FourCC Core::RefCounted::GetClassFourCC()` const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

`void Core::RefCounted::DumpRefCountingLeaks()` [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::GameStateHandler
BaseGameFeature::GameStateHandler
Class Reference

#include <gamestatehandler.h>
Detailed Description

The **GameStateHandler** creates the basegamefeature and attached it to the gameserver. It uses the basegamefeature to load a level, a savegame or setups a new game or just an empty world.

The basegamefeature does the specific game logic stuff, like handling entities with the entitymanager and factorymanager and so on.

(C) 2003 RadonLabs GmbH
Public Types

enum SetupMode
  setup modes
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GameStateHandler ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~GameStateHandler ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual void OnStateEnter (const Util::String &amp;prevState)</strong></td>
<td>called when the state represented by this state handler is entered</td>
</tr>
<tr>
<td><strong>virtual void OnStateLeave (const Util::String &amp;nextState)</strong></td>
<td>called when the state represented by this state handler is left</td>
</tr>
<tr>
<td><strong>virtual Util::String OnFrame ()</strong></td>
<td>called each frame as long as state is current, return new state</td>
</tr>
<tr>
<td><strong>void SetUpMode (SetupMode mode)</strong></td>
<td>set the setup mode</td>
</tr>
<tr>
<td><strong>SetupMode GetSetUpMode () const</strong></td>
<td>get the setup mode</td>
</tr>
<tr>
<td><strong>void SetLevelName (const Util::String &amp;n)</strong></td>
<td>set level filename, required by setup mode LoadLevel</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetLevelName () const</strong></td>
<td>get level name</td>
</tr>
<tr>
<td><strong>void SetSaveGame (const Util::String &amp;n)</strong></td>
<td>set save game name, required by setup mode LoadSaveGame</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetSaveGame () const</strong></td>
<td>get save game name</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void BaseGameFeature::GameStateHandler::OnStateEnter ( const Util::String prevState ) [virtual]

called when the state represented by this state handler is entered

This method is called when the state associated with this state handler is entered. The parameter indicates the previously active state.

Parameters:
  prevState  previous state
```

```cpp
void BaseGameFeature::GameStateHandler::OnStateLeave ( const Util::String nextState ) [virtual]

called when the state represented by this state handler is left

This method is called when the state associated with this state handler is left. The parameter indicates the next active state.

Parameters:
  nextState  next state
```

```cpp
Util::String BaseGameFeature::GameStateHandler::OnFrame ( ) [virtual]

called each frame as long as state is current, return new state

This method is called once a frame while the state is active. The method must return a state identifier. If this is different from the current state, a state switch will occur after the method returns.

Returns:
  a state identifier
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**BaseGameFeature::GlobalAttrsManager**
BaseGameFeature::GlobalAttrsManager
Class Reference

#include <globalattrsmanager.h>

Inheritance diagram for BaseGameFeature::GlobalAttrsManager:
Detailed Description

Provides read/write access to global attributes.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GlobalAttrsManager ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~GlobalAttrsManager ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void OnActivate ()</td>
<td>called when attached to game server</td>
</tr>
<tr>
<td>void SaveAttributes ()</td>
<td>explicitly save attributes to database (not through OnSave!)</td>
</tr>
<tr>
<td>void LoadAttributes ()</td>
<td>explicitly load attributes from database (not through OnLoad())</td>
</tr>
<tr>
<td>bool HasAttr (const Attr::AttrId &amp;attrId) const</td>
<td>return true if global attribute exists</td>
</tr>
<tr>
<td>void SetString (const Attr::StringAttrId &amp;attrId, const Util::String &amp;value)</td>
<td>set a global string attribute</td>
</tr>
<tr>
<td>const Util::String &amp; GetString (const Attr::StringAttrId &amp;attrId) const</td>
<td>get a global string attribute</td>
</tr>
<tr>
<td>void SetInt (const Attr::IntAttrId &amp;attrId, int &amp;value)</td>
<td>set a global int attribute</td>
</tr>
<tr>
<td>int GetInt (const Attr::IntAttrId &amp;attrId) const</td>
<td>get a global int attribute</td>
</tr>
<tr>
<td>void SetFloat (const Attr::FloatAttrId &amp;attrId, float &amp;value)</td>
<td>set a global float attribute</td>
</tr>
<tr>
<td>float GetFloat (const Attr::FloatAttrId &amp;attrId) const</td>
<td>get a global float attribute</td>
</tr>
<tr>
<td>void SetBool (const Attr::BoolAttrId &amp;attrId, bool &amp;value)</td>
<td>set a global bool attribute</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>bool</td>
<td><code>GetBool</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetFloat4</code></td>
</tr>
<tr>
<td>const <code>Math::float4</code></td>
<td><code>GetFloat4</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetMatrix44</code></td>
</tr>
<tr>
<td>const <code>Math::matrix44</code></td>
<td><code>GetMatrix44</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetGuid</code></td>
</tr>
<tr>
<td>const <code>Util::Guid</code> &amp;</td>
<td><code>GetGuid</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetBlob</code></td>
</tr>
<tr>
<td>const <code>Util::Blob</code> &amp;</td>
<td><code>GetBlob</code></td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnDeactivate</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsActive</code></td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnBeginFrame</code></td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnFrame</code></td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnEndFrame</code></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void OnLoad ()</td>
<td>called after loading game state</td>
</tr>
<tr>
<td>virtual void OnSave ()</td>
<td>called before saving game state</td>
</tr>
<tr>
<td>virtual void OnStart ()</td>
<td>called by Game::Server::Start() when the world is started</td>
</tr>
<tr>
<td>virtual void OnRenderDebug ()</td>
<td>render a debug visualization</td>
</tr>
<tr>
<td>virtual void HandleMessage (const Ptr&lt; Messaging::Message &gt; &amp;msg)</td>
<td>handle a single message (distribute to ports which accept the message)</td>
</tr>
<tr>
<td>void AttachPort (const Ptr&lt; Port &gt; &amp;port)</td>
<td>attach a message port</td>
</tr>
<tr>
<td>void RemovePort (const Ptr&lt; Port &gt; &amp;port)</td>
<td>remove a message port</td>
</tr>
<tr>
<td>bool HasPort (const Ptr&lt; Port &gt; &amp;port) const</td>
<td>return true if a port exists</td>
</tr>
<tr>
<td>virtual void SetupAcceptedMessages ()</td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>void AttachHandler (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td>void RemoveHandler (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td>void RemoveAllHandlers ()</td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td>SizeT GetNumHandlers () const</td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td>const Ptr&lt; Handler &gt; &amp; GetHandlerAtIndex (IndexT i) const</td>
<td>get a message handler by index</td>
</tr>
<tr>
<td>virtual void Send (const Ptr&lt; Message &gt; &amp;msg)</td>
<td>send a message to the port</td>
</tr>
<tr>
<td>const Util::Array&lt; const Id * &gt; &amp; GetAcceptedMessages () const</td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td>bool AcceptsMessage (const Id &amp;msgId) const</td>
<td></td>
</tr>
</tbody>
</table>
return true if port accepts this msg

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

```c++
void RegisterMessage (const Id &msgId)
    register a single accepted message
```
Member Function Documentation

void
Game::Manager::OnDeactivate( ) [virtual, inherited]

called when removed from game server

This method is called when the manager is removed from the game server. It will unregister its message port from the message server at this point.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager, BaseGameFeature::EntityManager, BaseGameFeature::EnvEntityManager, BaseGameFeature::EnvQueryManager, Script::DialogManager, and Script::ScriptManager.

void
Game::Manager::OnBeginFrame( ) [virtual, inherited]

called before frame by the game server

Called before frame, override in subclasses

Reimplemented in BaseGameFeature::EntityManager.

void
Game::Manager::OnEndFrame( ) [virtual, inherited]

called after frame by the game server

Called after frame, override in subclasses

Reimplemented in BaseGameFeature::EntityManager.

void
Messaging::Dispatcher::HandleMessage( const Ptr<Messaging::Message> &msg ) [virtual, inherited]
handle a single message (distribute to ports which accept the message)

Handle a message. The message will only be distributed to ports which accept the message.

Reimplemented from **Messaging::Port**.

Reimplemented in **Script::DialogManager**.

```cpp
void Messaging::Dispatcher::AttachPort(
    const Ptr<Port> & port)
[inherited]
```

**attach a message port**

Attach a new message port.

**Parameters:**

- `port` pointer to a message port object

```cpp
void Messaging::Dispatcher::RemovePort(
    const Ptr<Port> & port)
[inherited]
```

**remove a message port**

Remove a message port object.

**Parameters:**

- `handler` pointer to message port object to be removed

```cpp
bool Messaging::Dispatcher::HasPort(
    const Ptr<Port> & port) const
[inherited]
```

**return true if a port exists**

Return true if a port is already attached.
attach a message handler to the port

Attach a message handler to the port.

remove a message handler from the port

Remove a message handler from the port.

send a message to the port

Send a message to the port. This will immediately call the HandleMessage() method of all attached handlers. If the message has been handled by at least one of the handlers, the Handled() flag of the message will be set to true.

get the current refcount

Return the current refcount of the object.

increment refcount by one

Increment the refcount of the object.
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::InputTimeSource
BaseGameFeature::InputTimeSource
Class Reference

#include <inputtimesource.h>

Inheritance diagram for BaseGameFeature::InputTimeSource:
Detailed Description

Provides a time source for input handling. This allows the Timing world to be paused independently from input handling.

(C) 2007 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InputTimeSource ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~InputTimeSource ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual void OnActivate ()</code></td>
<td>called when time source is attached to the time manager</td>
</tr>
<tr>
<td><code>virtual void OnDeactivate ()</code></td>
<td>called when time source is removed from the time manager</td>
</tr>
<tr>
<td><code>virtual void OnLoad (const Ptr&lt; Db::Reader &gt; &amp;dbReader)</code></td>
<td>read state from database reader</td>
</tr>
<tr>
<td><code>virtual void OnSave (const Ptr&lt; Db::Writer &gt; &amp;dbWriter)</code></td>
<td>write state to database writer</td>
</tr>
<tr>
<td><code>virtual void Reset ()</code></td>
<td>reset the time source to 0.0</td>
</tr>
<tr>
<td><code>virtual void Pause ()</code></td>
<td>pause the time source (increments pause counter)</td>
</tr>
<tr>
<td><code>virtual void Continue ()</code></td>
<td>unpause the time source (decrements pause counter)</td>
</tr>
<tr>
<td><code>bool IsPaused () const</code></td>
<td>return true if currently paused</td>
</tr>
<tr>
<td><code>virtual void SetFactor (float f)</code></td>
<td>set acceleration/deceleration factor</td>
</tr>
<tr>
<td><code>float GetFactor () const</code></td>
<td>get acceleration/deceleration factor</td>
</tr>
<tr>
<td><code>Timing::Time GetTime () const</code></td>
<td>get the current time</td>
</tr>
<tr>
<td><code>Timing::Time GetFrameTime () const</code></td>
<td>get the current frame time</td>
</tr>
<tr>
<td><code>uint GetFrameId () const</code></td>
<td>get a unique frame id</td>
</tr>
<tr>
<td><code>Timing::Tick GetTicks () const</code></td>
<td>get current ticks</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td></td>
</tr>
</tbody>
</table>
get the current refcount

void **AddRef ()
increment refcount by one

void **Release ()
decrement refcount and destroy object if refcount is zero

bool **IsInstanceOf (const **Rtti &rtti) const
return true if this object is instance of given class

bool **IsInstanceOf (const **Util::String &className) const
return true if this object is instance of given class by string

bool **IsInstanceOf (const **Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool **IsA (const **Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool **IsA (const **Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool **IsA (const **Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const **Util::String & **GetClassName () const
get the class name

**Util::FourCC **GetClassFourCC () const
get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

```
virtual void UpdateTime (Timing::Time frameTime, Timing::Tick t)
update current time (called by time manager)
```
Member Function Documentation

```cpp
void
BaseGameFeature::TimeSource::UpdateTime
(Timing::Time frameTime,
 Timing::Tick t)
[protected, virtual, inherited]
```

update current time (called by time manager)

This method is called when the internal time should be updated. It will be called early in the frame by the `TimeManager`. Think of this method as the time source's `OnFrame()` method.

```cpp
int
Core::RefCounted::GetRefCount()
const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void
Core::RefCounted::AddRef()
[inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release()
[inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName()
const [inline, inherited]
```

get the class name
Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::LevelLoader
BaseGameFeature::LevelLoader Class Reference

#include <levelloader.h>
Detailed Description

Helper class for loading a complete level from the world database.

(C) 2007 Radon Labs GmbH
Static Public Member Functions

```cpp
static bool Load (const Util::String &levelName)
load a complete level from the world database
```

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:41 2010
BaseGameFeature::LoaderServer
BaseGameFeature::LoaderServer Class Reference

#include <loaderserver.h>

Inheritance diagram for BaseGameFeature::LoaderServer:
Detailed Description

The BaseGameFeature::LoaderServer is the central object of the loader subsystem. Usually you don't work directly with the Loader subsystem, but instead use higher level classes like the Game::SetupManager and Game::SaveGameManager.

(C) 2003 RadonLabs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LoaderServer()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~LoaderServer()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void SetDebugTextEnabled(bool b)</code></td>
<td>enable/disable debug text messages during load</td>
</tr>
<tr>
<td><code>bool GetDebugTextEnabled()</code></td>
<td>get debug text enabled flag</td>
</tr>
<tr>
<td><code>virtual bool Open()</code></td>
<td>open the loader subsystem</td>
</tr>
<tr>
<td><code>virtual void Close()</code></td>
<td>close the loader subsystem</td>
</tr>
<tr>
<td><code>bool IsOpen()</code></td>
<td>return true if open</td>
</tr>
<tr>
<td><code>virtual Ptr&lt;UserProfile&gt; CreateUserProfile()</code></td>
<td>create a new user profile object</td>
</tr>
<tr>
<td><code>void SetUserProfile(const Ptr&lt;UserProfile&gt; &amp;p)</code></td>
<td>set the current user profile</td>
</tr>
<tr>
<td><code>const Ptr&lt;UserProfile&gt; &amp; GetUserProfile()</code></td>
<td>get the current user profile</td>
</tr>
<tr>
<td><code>virtual bool LoadLevel(const Util::String &amp;levelName)</code></td>
<td>load a new level, this method is usually called by Game::SetupManager</td>
</tr>
</tbody>
</table>
| `void AttachEntityLoader(const Ptr<
  EntityLoaderBase> &loader)` | attach loader                                                               |
| `void RemoveEntityLoader(const Ptr<
  EntityLoaderBase> &loader)` | remove loader                                                               |
| `void RemoveAllLoaders()`                    | remove all loaders                                                          |
| `void LoadEntities(const Util::Array<
  Util::String> &activeLayers)` | load entities                                                               |
<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <strong>SetProgressResource</strong> (const <strong>Util::String</strong> &amp;r)</td>
<td>set progress indicator gui resource</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetProgressResource</strong> () const</td>
<td>get progress indicator gui resource</td>
</tr>
<tr>
<td>void <strong>SetMaxProgressValue</strong> (int v)</td>
<td>set the max progress value</td>
</tr>
<tr>
<td>int <strong>GetMaxProgressValue</strong> () const</td>
<td>get the max progress value</td>
</tr>
<tr>
<td>void <strong>AdvanceProgress</strong> (int amount)</td>
<td>advance the progress indicator</td>
</tr>
<tr>
<td>void <strong>SetProgressText</strong> (const <strong>Util::String</strong> &amp;s)</td>
<td>set optional progress text</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetProgressText</strong> () const</td>
<td>get optional progress text</td>
</tr>
<tr>
<td>virtual void <strong>UpdateProgressDisplay</strong> ()</td>
<td>update the progress indicator display</td>
</tr>
<tr>
<td>void <strong>OpenProgressIndicator</strong> ()</td>
<td>open the progress indicator</td>
</tr>
<tr>
<td>void <strong>CloseProgressIndicator</strong> ()</td>
<td>close the progress indicator</td>
</tr>
<tr>
<td><strong>IO::URI</strong> <strong>GetScreenshotFilepath</strong> (const <strong>Util::String</strong> &amp;extension)</td>
<td>get filename for next screenshot</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td></td>
<td>or a derived class</td>
</tr>
<tr>
<td><code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class,</td>
</tr>
<tr>
<td></td>
<td>or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class,</td>
</tr>
<tr>
<td></td>
<td>or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool BaseGameFeature::LoaderServer::Open() [virtual]
```

open the loader subsystem

Open the loader subsystem.

**Parameters:**

- `applicationName` the app name
- `fileFormatVersion` string in the form "x.y" describing the file format version

**Returns:**

true if loader subsystem successfully opened

```cpp
void BaseGameFeature::LoaderServer::Close() [virtual]
```

close the loader subsystem

Close the loader subsystem.

```cpp
Ptr<UserProfile> BaseGameFeature::LoaderServer::CreateUserProfile() const [virtual]
```

create a new user profile object

Creates a new user profile object. Override in subclass to create your own user profile subclass instances.

```cpp
void BaseGameFeature::LoaderServer::SetUserProfile(const Ptr<UserProfile> & p) [inline]
```

set the current user profile

This sets the current user profile.
get the current user profile

Returns the current user profile

```cpp
bool BaseGameFeature::LoaderServer::LoadLevel ( const Util::String levelName ) [virtual]
```

load a new level, this method is usually called by Game::SetupManager

Load a new game level from the world database.

**Parameters:**
- `filename` a level name (this is not a filename!)

**Returns:**
- `success`

```cpp
void BaseGameFeature::LoaderServer::AttachEntityLoader ( const Ptr<EntityLoaderBase> & loader )
```

attach loader

Attach new entity loader to **LoaderServer**.

**Parameters:**
- `loader` loader class

```cpp
void BaseGameFeature::LoaderServer::RemoveEntityLoader ( const Ptr<EntityLoaderBase> & loader )
```

remove loader

Remove given loader from **LoaderServer**.

**Parameters:**
- `loader` loader class

```cpp
void BaseGameFeature::LoaderServer::RemoveAllLoaders ()
```
remove all loaders

Remove all loaders.

```cpp
void BaseGameFeature::LoaderServer::LoadEntities(const Util::Array<Util::String>& activeLayers)
```

load entities from db with entityloader

Go thru all entity loader and call its Load function.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String& Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC const [inline, inherited]
```
Core::RefCounted::GetClassFourCC()

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks()

[static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::SystemTimeSource
BaseGameFeature::SystemTimeSource
Class Reference

#include <systemtimesource.h>

Inheritance diagram for BaseGameFeature::SystemTimeSource:
Detailed Description

Provides the time for the various system features. These features has to get the time from this timesource.

Access the SystemTimeSource object as Singleton:

Time sysTime = SystemTimeSource::Instance()->GetTime();

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SystemTimeSource</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~SystemTimeSource</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <strong>OnActivate</strong> ()</td>
<td>called when time source is attached to the time manager</td>
</tr>
<tr>
<td>virtual void <strong>OnDeactivate</strong> ()</td>
<td>called when time source is removed from the time manager</td>
</tr>
<tr>
<td>virtual void <strong>OnLoad</strong> (const <strong>Ptr</strong><a href="">Db::Reader</a>&amp; dbReader)</td>
<td>read state from database reader</td>
</tr>
<tr>
<td>virtual void <strong>OnSave</strong> (const <strong>Ptr</strong><a href="">Db::Writer</a>&amp; dbWriter)</td>
<td>write state to database writer</td>
</tr>
<tr>
<td>virtual void <strong>Reset</strong> ()</td>
<td>reset the time source to 0.0</td>
</tr>
<tr>
<td>virtual void <strong>Pause</strong> ()</td>
<td>pause the time source (increments pause counter)</td>
</tr>
<tr>
<td>virtual void <strong>Continue</strong> ()</td>
<td>unpause the time source (decrements pause counter)</td>
</tr>
<tr>
<td>bool <strong>IsPaused</strong> () const</td>
<td>return true if currently paused</td>
</tr>
<tr>
<td>virtual void <strong>SetFactor</strong> (float f)</td>
<td>set acceleration/deceleration factor</td>
</tr>
<tr>
<td>float <strong>GetFactor</strong> () const</td>
<td>get acceleration/deceleration factor</td>
</tr>
<tr>
<td>Timing::Time <strong>GetTime</strong> () const</td>
<td>get the current time</td>
</tr>
<tr>
<td>Timing::Time <strong>GetFrameTime</strong> () const</td>
<td>get the current frame time</td>
</tr>
<tr>
<td>uint <strong>GetFrameId</strong> () const</td>
<td>get a unique frame id</td>
</tr>
<tr>
<td>Timing::Tick <strong>GetTicks</strong> () const</td>
<td>get current ticks</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

virtual void UpdateTime (Timing::Time frameTime, Timing::Tick t)
update current time (called by time manager)
Member Function Documentation

void
BaseGameFeature::TimeSource::UpdateTime (Timing::Time frameTime,
    Timing::Tick t)
    [protected, virtual, inherited]

update current time (called by time manager)

This method is called when the internal time should be updated. It will be called early in the frame by the TimeManager. Think of this method as the time source's OnFrame() method.

int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Utility::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name
Get the class name of the object.

\texttt{Util::FourCC}
\texttt{Core::RefCounted::GetClassFourCC() const [inline, inherited]}

get the class FourCC code

Get the class FourCC of the object.

\texttt{void}
\texttt{Core::RefCounted::DumpRefCountingLeaks() [static, inherited]}

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::TimeManager
BaseGameFeature::TimeManager Class Reference

#include <timemanager.h>

Inheritance diagram for BaseGameFeature::TimeManager:
Detailed Description

Singleton object which manages the current game time. These are the standard time source objects provided by Application layer:

**SystemTimeSource** - timing for low level Application layer subsystems
**GameTimeSource** - timing for the game logic
**CameraTimeSource** - extra time source for camera handling
**GuiTimeSource** - time source for user interface stuff

The **TimeManager** offers a TimeEffect for animating the timefactor of all time sources AND of the graphicsthread to allow "Matrix"-like time effects.

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TimeManager</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~TimeManager</strong> ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>OnActivate</strong> ()</td>
<td>Called when attached to game server</td>
</tr>
<tr>
<td>virtual void <strong>OnDeactivate</strong> ()</td>
<td>Called when removed from game server</td>
</tr>
<tr>
<td>virtual void <strong>OnLoad</strong> ()</td>
<td>Called after loading game state</td>
</tr>
<tr>
<td>virtual void <strong>OnSave</strong> ()</td>
<td>Called before saving game state</td>
</tr>
<tr>
<td>virtual void <strong>OnFrame</strong> ()</td>
<td>On frame</td>
</tr>
<tr>
<td>void <strong>ResetAll</strong> ()</td>
<td>Reset all time sources</td>
</tr>
<tr>
<td>void <strong>PauseAll</strong> ()</td>
<td>Pause all time sources</td>
</tr>
<tr>
<td>void <strong>ContinueAll</strong> ()</td>
<td>Continue all time sources</td>
</tr>
<tr>
<td>void <strong>SetTimeFactor</strong> (float timeFactor, bool resetTimeEffects=true)</td>
<td>Set the time factor</td>
</tr>
<tr>
<td>float <strong>GetTimeFactor</strong> () const</td>
<td>Get the time factor</td>
</tr>
<tr>
<td>virtual void <strong>OnRenderDebug</strong> ()</td>
<td>Render a debug visualization</td>
</tr>
<tr>
<td>void <strong>AttachTimeSource</strong> (const <em>TimeSource</em> &amp;timeSource)</td>
<td>Attach a time source</td>
</tr>
<tr>
<td>void <strong>RemoveTimeSource</strong> (const <em>TimeSource</em> &amp;timeSource)</td>
<td>Remove a time source</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>int GetNumTimeSources() const</code></td>
<td>Get number of time source</td>
</tr>
<tr>
<td><code>const Ptr&lt; TimeSource &gt; &amp; GetTimeSourceByIndex(int index) const</code></td>
<td>Get pointer to time source by index</td>
</tr>
<tr>
<td><code>Ptr&lt; TimeSource &gt; GetTimeSourceByClassName(const Util::String &amp;n) const</code></td>
<td>Get pointer to time source by class name</td>
</tr>
<tr>
<td><code>void StartTimeEffect(float timeFactor, Timing::Tick duration, Timing::Tick fadeIn, Timing::Tick fadeOut)</code></td>
<td>Start a time effect to animate time factor (&quot;Matrix/MaxPayne&quot; time effect)</td>
</tr>
<tr>
<td><code>void StopTimeEffect(bool immediate=false)</code></td>
<td>Stop current time effect</td>
</tr>
<tr>
<td><code>bool IsTimeEffectActive() const</code></td>
<td>Is time effect active</td>
</tr>
<tr>
<td><code>bool IsActive() const</code></td>
<td>Return true if currently active</td>
</tr>
<tr>
<td><code>virtual void OnBeginFrame()</code></td>
<td>Called before frame by the game server</td>
</tr>
<tr>
<td><code>virtual void OnEndFrame()</code></td>
<td>Called after frame by the game server</td>
</tr>
<tr>
<td><code>virtual void OnStart()</code></td>
<td>Called by Game::Server::Start() when the world is started</td>
</tr>
<tr>
<td><code>virtual void HandleMessage(const Ptr&lt;Messaging::Message&gt; &amp;msg)</code></td>
<td>Handle a single message (distribute to ports which accept the message)</td>
</tr>
<tr>
<td><code>void AttachPort(const Ptr&lt; Port &gt; &amp;port)</code></td>
<td>Attach a message port</td>
</tr>
<tr>
<td><code>void RemovePort(const Ptr&lt; Port &gt; &amp;port)</code></td>
<td>Remove a message port</td>
</tr>
<tr>
<td><code>bool HasPort(const Ptr&lt; Port &gt; &amp;port) const</code></td>
<td>Return true if a port exists</td>
</tr>
<tr>
<td><code>virtual void SetupAcceptedMessages()</code></td>
<td>Override to register accepted messages</td>
</tr>
<tr>
<td><code>void AttachHandler(const Ptr&lt; Handler &gt; &amp;h)</code></td>
<td>Attach a message handler</td>
</tr>
<tr>
<td>Function/Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void RemoveHandler (const Ptr&lt; Handler &gt;&amp; h)</code></td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td><code>void RemoveAllHandlers ()</code></td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td><code>SizeT GetNumHandlers () const</code></td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td><code>const Ptr&lt; Handler &gt;&amp; GetHandlerAtIndex (IndexT i)</code> const</td>
<td>get a message handler by index</td>
</tr>
<tr>
<td><code>virtual void Send (const Ptr&lt; Message &gt;&amp; msg)</code></td>
<td>send a message to the port</td>
</tr>
<tr>
<td><code>const Util::Array&lt; const Id * &gt;&amp; GetAcceptedMessages () const</code></td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td><code>bool AcceptsMessage (const Id &amp;msgId)</code> const</td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void ApplyGlobalAudioPitch()</td>
<td>const apply the audio variable</td>
</tr>
<tr>
<td>void Update()</td>
<td>update the time manager</td>
</tr>
<tr>
<td>void ApplyTimeEffect()</td>
<td>apply time effect</td>
</tr>
<tr>
<td>void RegisterMessage()</td>
<td>(const Id &amp;msgId) register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

void BaseGameFeature::TimeManager::OnActivate( ) [virtual]
called when attached to game server
Activate the time manager. This will create all the standard time sources for Mangalore.
Reimplemented from Game::Manager.

void BaseGameFeature::TimeManager::OnDeactivate( ) [virtual]
called when removed from game server
Deactivate the time manager.
Reimplemented from Game::Manager.

void BaseGameFeature::TimeManager::OnLoad( ) [virtual]
called after loading game state
Checks whether the TimeSources table exists in the database, if yes invokes OnLoad() on all time sources...
Reimplemented from Game::Manager.

void BaseGameFeature::TimeManager::OnSave( ) [virtual]
called before saving game state
Ask all time sources to save their status to the database.
Reimplemented from Game::Manager.
void BaseGameFeature::TimeManager::OnFrame()

on frame

Update our time sources

Reimplemented from Game::Manager.

void BaseGameFeature::TimeManager::ResetAll()

reset all time sources

Reset all time sources. This is usually called at the beginning of an application state.

void BaseGameFeature::TimeManager::PauseAll()

pause all time sources

Pause all time sources. NOTE: there's an independent pause counter inside each time source, a pause just increments the counter, a continue decrements it, when the pause counter is != 0 it means, pause is activated.

void BaseGameFeature::TimeManager::ContinueAll()

continue all time sources

Unpause all time sources.

void BaseGameFeature::TimeManager::OnRenderDebug()

render a debug visualization

show time factor if not 1.0

Reimplemented from Game::Manager.
void BaseGameFeature::TimeManager::AttachTimeSource(const Ptr<
    TimeSource> timeSource &)

attach a time source

Attach a time source to the time manager. This will invoke
\texttt{OnActivate()} on the time source.

void BaseGameFeature::TimeManager::RemoveTimeSource(const
    Ptr<
    TimeSource> timeSource &)

remove a time source

Remove a time source from the time manager. This will invoke
\texttt{OnDeactivate()} on the time source.

int BaseGameFeature::TimeManager::GetNumTimeSources()
  const

get number of time source

Returns number of time sources attached to the time manager.

const Ptr< TimeSource > &
BaseGameFeature::TimeManager::GetTimeSourceByIndex(int index)
  const

get pointer to time source by index

Gets pointer to time source object by index.

\texttt{Ptr< TimeSource >}
BaseGameFeature::TimeManager::GetTimeSourceByClassName
    (const 
    Util::String \texttt{n} )
  const

get pointer to time source by class name

Get pointer to time source object by class name, returns 0 if not found.

void BaseGameFeature::TimeManager::Update()
  [protected]
update the time manager

Update all time sources. This method is called very early in the frame by the current application state handler. This will get the current frame time and call UpdateTime() on all attached time sources.

FIXME: properly handle time exceptions!!

```cpp
void Game::Manager::OnBeginFrame() [virtual, inherited]
```
called before frame by the game server

Called before frame, override in subclasses

Reimplemented in BaseGameFeature::EntityManager.

```cpp
void Game::Manager::OnEndFrame() [virtual, inherited]
```
called after frame by the game server

Called after frame, override in subclasses

Reimplemented in BaseGameFeature::EntityManager.

```cpp
void Messaging::Dispatcher::HandleMessage(const Ptr<Messaging::Message> &msg) [virtual, inherited]
```
handle a single message (distribute to ports which accept the message)

Handle a message. The message will only be distributed to ports which accept the message.

Reimplemented from Messaging::Port.

Reimplemented in Script::DialogManager.
attach a message port

Attach a new message port.

**Parameters:**

- `port` pointer to a message port object

```cpp
void Messaging::Dispatcher::AttachPort ( Port port )
```

remove a message port

Remove a message port object.

**Parameters:**

- `handler` pointer to message port object to be removed

```cpp
void Messaging::Dispatcher::RemovePort ( const_ptr< Port > & port ) [inherited]
```

return true if a port exists

Return true if a port is already attached.

```cpp
bool Messaging::Dispatcher::HasPort ( const_ptr< Port > & port ) const [inherited]
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::AttachHandler ( const_ptr< Handler > & h ) [inherited]
```

```cpp
void Messaging::Port::RemoveHandler ( const_ptr< Handler > & h ) [inherited]
```
remove a message handler from the port

Remove a message handler from the port.

```cpp
void Messaging::Port::Send(const Ptr<Message> &msg) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name
Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::TimeProperty
BaseGameFeature::TimeProperty
Class Reference

#include <timeproperty.h>
Detailed Description

The time property adds the attribute "Time" to the entity. This attribute contains the time since the time property has been attached to the entity.

(C) 2007 Radon Labs GmbH
BaseGameFeature::TimeSource
#include <timesource.h>

Inheritance diagram for BaseGameFeature::TimeSource:
Detailed Description

A generic time source object which is attached to the TimeManager. Each time source tracks its own time independently from the other time sources, they can also be paused and unpaved independently from each other, and they may also run faster or slower than realtime.

To create new time sources, derive a subclass from time source. This is necessary because time source objects are identified by their class id. A positive side effect of this is that time sources are created as singletons, so access is very simple, e.g.:

\[
\text{Time gameTime} = \text{GameTime::Instance() -> \text{GetTime}();}
\]

Time source classes are responsible for loading/saving their state into the globals attribute table.

All timesources have to be passiv, everyone who needs time has to get it on its own from one of the different timsources. No time source should set the time in a subsystem activly.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TimeSource()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~TimeSource()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>OnActivate()</code></td>
<td>Called when time source is attached to the time manager</td>
</tr>
<tr>
<td><code>OnDeactivate()</code></td>
<td>Called when time source is removed from the time manager</td>
</tr>
<tr>
<td><code>OnLoad(const Ptr&lt;Db::Reader&gt;&amp; dbReader)</code></td>
<td>Read state from database reader</td>
</tr>
<tr>
<td><code>OnSave(const Ptr&lt;Db::Writer&gt;&amp; dbWriter)</code></td>
<td>Write state to database writer</td>
</tr>
<tr>
<td><code>Reset()</code></td>
<td>Reset the time source to 0.0</td>
</tr>
<tr>
<td><code>Pause()</code></td>
<td>Pause the time source (increments pause counter)</td>
</tr>
<tr>
<td><code>Continue()</code></td>
<td>Unpause the time source (decrements pause counter)</td>
</tr>
<tr>
<td><code>IsPaused() const</code></td>
<td>Return true if currently paused</td>
</tr>
<tr>
<td><code>SetFactor(float f)</code></td>
<td>Set acceleration/deceleration factor</td>
</tr>
<tr>
<td><code>GetFactor() const</code></td>
<td>Get acceleration/deceleration factor</td>
</tr>
<tr>
<td><code>Timing::Time GetTime() const</code></td>
<td>Get the current time</td>
</tr>
<tr>
<td><code>Timing::Time GetFrameTime() const</code></td>
<td>Get the current frame time</td>
</tr>
<tr>
<td><code>uint GetFrameId() const</code></td>
<td>Get a unique frame id</td>
</tr>
<tr>
<td><code>Timing::Tick GetTicks() const</code></td>
<td>Get current ticks</td>
</tr>
<tr>
<td><code>int GetRefCount() const</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Protected Member Functions

virtual void **UpdateTime** (Timing::Time frameTime, Timing::Tick t)

*update current time (called by time manager)*
**Member Function Documentation**

```cpp
void BaseGameFeature::TimeSource::UpdateTime(Timing::Time frameTime,
                                            Timing::Tick t) [protected, virtual]
```

update current time (called by time manager)

This method is called when the internal time should be updated. It will be called early in the frame by the `TimeManager`. Think of this method as the time source's `OnFrame()` method.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.
Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
BaseGameFeature::TimingTimeSource
BaseGameFeature::TimingTimeSource
Class Reference

#include <gametimesource.h>
Detailed Description

Provides timing information for the Timing logic.

(C) 2007 Radon Labs GmbH
BaseGameFeature::TransformableProperty
BaseGameFeature::TransformableProperty
Class Reference

#include <transformableproperty.h>

Inheritance diagram for BaseGameFeature::TransformableProperty:

- Core::RefCounted
- Messaging::Port
- Game::Property
- BaseGameFeature::TransformableProperty
- PhysicsFeature::PhysicsProperty
- PhysicsFeature::TriggerProperty
- PhysicsFeature::ActorPhysicsProperty
Detailed Description

Entities with this property can be transformed.

(C) 2007 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>CallbackType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>call backs</td>
</tr>
</tbody>
</table>

*callback types*
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TransformableProperty()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~TransformableProperty()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <code>SetupDefaultAttributes()</code></td>
<td>setup default entity attributes</td>
</tr>
<tr>
<td>virtual void <code>SetupAcceptedMessages()</code></td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>virtual void <code>HandleMessage(const Ptr&lt;Messaging::Message&gt;&amp; msg)</code></td>
<td>handle a single message</td>
</tr>
<tr>
<td>const <code>Ptr&lt;Entity&gt;</code> &amp; <code>GetEntity()</code></td>
<td>get entity this property is attached to</td>
</tr>
<tr>
<td>bool <code>HasEntity()</code></td>
<td>return true if entity pointer is valid</td>
</tr>
<tr>
<td>virtual void <code>SetupCallbacks()</code></td>
<td>setup callbacks for this property, call by entity in <code>OnActivate()</code></td>
</tr>
<tr>
<td>virtual void <code>OnActivate()</code></td>
<td>called from <code>Entity::activateProperties()</code></td>
</tr>
<tr>
<td>virtual void <code>OnDeactivate()</code></td>
<td>called from <code>Entity::DeactivateProperties()</code></td>
</tr>
<tr>
<td>bool <code>IsActive()</code></td>
<td>return true if property is currently active</td>
</tr>
<tr>
<td>virtual void <code>OnLoad()</code></td>
<td>called from within <code>Entity::Load()</code> after attributes are loaded</td>
</tr>
<tr>
<td>virtual void <code>OnStart()</code></td>
<td>called from within <code>Entity::OnStart()</code> after <code>OnLoad</code> when the complete world exist</td>
</tr>
<tr>
<td>virtual void <code>OnSave()</code></td>
<td>called from within <code>Entity::Save()</code> before attributes are saved back to database</td>
</tr>
<tr>
<td>virtual void <code>OnBeginFrame()</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>virtual void OnMoveBefore()</code></td>
<td>called on begin of frame</td>
</tr>
<tr>
<td><code>virtual void OnMoveAfter()</code></td>
<td>called before movement happens</td>
</tr>
<tr>
<td><code>virtual void OnRender()</code></td>
<td>called after movement has happened</td>
</tr>
<tr>
<td><code>virtual void OnRenderDebug()</code></td>
<td>called before rendering happens</td>
</tr>
<tr>
<td><code>virtual void OnLoseActivity()</code></td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td><code>virtual void OnGainActivity()</code></td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td><code>void AttachHandler(const Ptr&lt; Handler&gt; &amp;h)</code></td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td><code>void RemoveHandler(const Ptr&lt; Handler&gt; &amp;h)</code></td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td><code>void RemoveAllHandlers()</code></td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td><code>SizeT GetNumHandlers() const</code></td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td><code>const Ptr&lt; Handler&gt; &amp; GetHandlerAtIndex(IndexT i) const</code></td>
<td>get a message handler by index</td>
</tr>
<tr>
<td><code>virtual void Send(const Ptr&lt; Message&gt; &amp;msg)</code></td>
<td>send a message to the port</td>
</tr>
<tr>
<td><code>const Util::Array&lt;const Id*&gt; &amp; GetAcceptedMessages() const</code></td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td><code>bool AcceptsMessage(const Id &amp;msgId)</code></td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td><code>int GetRefCount() const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
**Protected Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetEntity (const Ptr&lt; Entity &gt; &amp;v)</code></td>
<td>Set entity, this is attached to, to <code>v</code>.</td>
</tr>
<tr>
<td><code>void ClearEntity ()</code></td>
<td>Remove entity.</td>
</tr>
<tr>
<td><code>void RegisterMessage (const Id &amp;msgId)</code></td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void Game::Property::SetupCallbacks() [virtual, inherited]
```

setup callbacks for this property, call by entity in **OnActivate()**

Tells the entity what per-frame callback methods should be called for this property. The method is called after **SetupDefaultAttributes()** by the entity, and the property is expected to call the **Entity::RegisterPropertyCallback()** once for every callback method (**OnBeginFrame()**, **OnMoveBefore()**, ...) that should be called per-frame.


```cpp
void Game::Property::OnActivate() [virtual, inherited]
```

called from **Entity::ActivateProperties()**

This method is called by **Game::Entity::ActivateProperties()**. Use this method for one-time initializations of the property.


```cpp
void
```
Game::Property::OnDeactivate() [virtual, inherited]

called from Entity::DeactivateProperties()

This method is called by Game::Entity::DeactivateProperties(). Use this method to cleanup stuff which has been initialized in OnActivate().


void Game::Property::OnLoad() [virtual, inherited]

called from within Entity::Load() after attributes are loaded

This method is called from within Game::Entity::Load() after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void Game::Property::OnStart() [virtual, inherited]

called from within Entity::OnStart() after OnLoad when the complete world exist

This method is called from within Game::Entity::OnStart(). This is the
moment when the world is complete and the entity can establish connections to other entities.

Reimplemented in **GraphicsFeature::CameraProperty**.

```cpp
void Game::Property::OnSave(); [virtual, inherited]
```
called from within Entity::Save() before attributes are saved back to database

This method is called from within Game::Entity::Save() before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in **PhysicsFeature::TriggerProperty**, and **StateObjectFeature::StateProperty**.

```cpp
void Game::Property::OnBeginFrame(); [virtual, inherited]
```
called on begin of frame

This method is called from Game::Entity::OnBeginFrame() on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in **PhysicsFeature::TriggerProperty**, and **StateObjectFeature::StateProperty**.

```cpp
void Game::Property::OnMoveBefore(); [virtual, inherited]
```
called before movement happens

This method is called from Game::Entity::OnMoveBefore() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before the physics
subsystem is triggered.

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`.

```cpp
void Game::Property::OnMoveAfter() [virtual, inherited]
```
called after movement has happened

This method is called from `Game::Entity::OnMoveAfter()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`, and `PhysicsFeature::PhysicsProperty`.

```cpp
void Game::Property::OnRender() [virtual, inherited]
```
called before rendering happens

This method is called from `Game::Entity::OnRender()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before rendering happens.

Reimplemented in `GraphicsFeature::CameraProperty`, `GraphicsFeature::ChaseCameraProperty`, and `GraphicsFeature::MayaCameraProperty`.

```cpp
void Game::Property::OnRenderDebug() [virtual, inherited]
```
called when game debug visualization is on

This method is called from `Game::Entity::OnRenderDebug()` on all properties attached to an entity in the order of attachment. It's meant for debug issues. It will be called when debug mode is enabled.

Reimplemented in `GraphicsFeature::GraphicsProperty`,
PhysicsFeature::ActorPhysicsProperty, and
PhysicsFeature::TriggerProperty.

```cpp
void Game::Property::OnLoseActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from Game::Entity::OnLoseActivity() on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

Reimplemented in PhysicsFeature::ActorPhysicsProperty.

```cpp
void Game::Property::OnGainActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from Game::Entity::OnRenderDebug() on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

```cpp
const Messaging::Port::AttachHandler(Ptr<Handler> & h) [inherited]
```
attach a message handler to the port

Attach a message handler to the port.

```cpp
const Messaging::Port::RemoveHandler(Handler & h) [inherited]
```
remove a message handler from the port
Remove a message handler from the port.

```cpp
void Messaging::Port::Send(const Ptr<Message> &msg ) [virtual, inherited]
```

Send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC ( ) const [inline, inherited]
```
Core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks()
[static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
BaseGameFeature::UserProfile
BaseGameFeature::UserProfile Class Reference

#include <userprofile.h>

Inheritance diagram for BaseGameFeature::UserProfile:
Detailed Description

A user profile represents a storage where all user specific data is kept across application restarts. This usually includes save games, options, and other per-user data. Mangalore applications should at least support a default profile, but everything is there to support more then one user profile.

User profiles are stored in "user:[appname]/profiles/[profilename]".

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>UserProfile ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~UserProfile ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>void <code>SetName (const Util::String &amp;n)</code></td>
<td>set the name of the user profile</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetName () const</code></td>
<td>get the name of the user profile</td>
</tr>
<tr>
<td>virtual void <code>SetToDefault ()</code></td>
<td>set the user profile to its default state, override in subclass</td>
</tr>
<tr>
<td>virtual bool <code>Load (const Util::String &amp;path=&quot;&quot;)</code></td>
<td>load the profile data</td>
</tr>
<tr>
<td>virtual bool <code>Save ()</code></td>
<td>save to disk</td>
</tr>
<tr>
<td>bool <code>IsLoaded () const</code></td>
<td>currently loaded?</td>
</tr>
<tr>
<td>Util::String <code>GetProfileDirectory () const</code></td>
<td>get the filesystem path to the user profile directory</td>
</tr>
<tr>
<td>Util::String <code>GetSaveGameDirectory () const</code></td>
<td>get the filesystem path to the savegame directory</td>
</tr>
<tr>
<td>Util::String <code>GetDatabasePath () const</code></td>
<td>get path to world database</td>
</tr>
<tr>
<td>Util::String <code>GetSaveGamePath (const Util::String &amp;saveGameName) const</code></td>
<td>get path to a complete savegame</td>
</tr>
<tr>
<td>bool <code>HasAttr (const Util::String &amp;name) const</code></td>
<td>return true if attribute exists in the profile</td>
</tr>
<tr>
<td>void <code>SetString (const Util::String &amp;name, const Util::String &amp;val)</code></td>
<td>set a string attribute in the profile</td>
</tr>
<tr>
<td>void <code>SetInt (const Util::String &amp;name, int val)</code></td>
<td>set an int attribute in the profile</td>
</tr>
<tr>
<td>void <code>SetFloat (const Util::String &amp;name, float val)</code></td>
<td>set a float attribute in the profile</td>
</tr>
</tbody>
</table>
set a float attribute in the profile

```cpp
void SetBool (const Util::String &name, bool val)
set a bool attribute in the profile
```

```cpp
void SetFloat4 (const Util::String &name, const Math::float4 &val)
set a float4 attribute in the profile
```

```
Util::String GetString (const Util::String &name) const
get string attribute from the profile
```

```cpp
int GetInt (const Util::String &name) const
get int attribute from the profile
```

```cpp
float GetFloat (const Util::String &name) const
get float attribute from the profile
```

```cpp
bool GetBool (const Util::String &name) const
get bool attribute from the profile
```

```
Math::float4 GetFloat4 (const Util::String &name) const
get float4 attribute from the profile
```

```cpp
int GetRefCount () const
get the current refcount
```

```cpp
void AddRef ()
increment refcount by one
```

```cpp
void Release ()
decrement refcount and destroy object if refcount is zero
```

```cpp
bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class
```

```cpp
bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class, by string
```

```cpp
bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class, by fourcc
```

```cpp
bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class
```

```cpp
bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string
```

```cpp
bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class,
| by fourcc |
|---|---|
| const **Util::String** & **GetClassName** () const |
| get the class name |
| **Util::FourCC** **GetClassFourCC** () const |
| get the class FourCC code |
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Util::Array&lt;Util::String&gt;</code></td>
<td><code>EnumProfiles()</code></td>
<td>static method to enumerate all existing user profiles</td>
</tr>
<tr>
<td>static void</td>
<td><code>DeleteProfile(const Util::String &amp;name)</code></td>
<td>static method to delete an existing user profile by name</td>
</tr>
<tr>
<td>static <code>Util::String</code></td>
<td><code>GetProfileRootDirectory()</code></td>
<td>static method which returns a path to the profile root directory</td>
</tr>
<tr>
<td>static void</td>
<td><code>DumpRefCountingLeaks()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Util::Array< Util::String >**

*BaseGameFeature::UserProfile::EnumProfiles ( ) [static]*

static method to enumerate all existing user profiles

This is a static method which returns the names of all user profiles which currently exist on disk.

```cpp
void
BaseGameFeature::UserProfile::DeleteProfile ( const Util::String & name ) [static]
```

static method to delete an existing user profile by name

This static method deletes an existing user profile on disk.

**Util::String**

*BaseGameFeature::UserProfile::GetProfileRootDirectory ( ) [static]*

static method which returns a path to the profile root directory

This static method returns the path to the profiles root directory for this application.

```cpp
void
BaseGameFeature::UserProfile::SetToDefault ( ) [virtual]
```

set the user profile to its default state, override in subclass

Set the user profile to its default state. This is empty in the base class but should be overridden to something meaningful in application specific subclasses.

```cpp
bool
BaseGameFeature::UserProfile::Load ( const Util::String & path = "" ) [virtual]
```

load the profile data
Load the profile data from disk file.

```cpp
bool BaseGameFeature::UserProfile::Save() [virtual]
```

save to disk

Close the profile. This will save the profile back to disc.

```cpp
Util::String BaseGameFeature::UserProfile::GetProfileDirectory() const
```

get the filesystem path to the user profile directory

Returns the path to the user's profile directory using the Nebula3 filesystem path conventions.

```cpp
Util::String BaseGameFeature::UserProfile::GetSaveGameDirectory() const
```

get the filesystem path to the savegame directory

Returns the path to the user's savegame directory (inside the profile directory) using the Nebula3 filesystem path conventions.

```cpp
Util::String BaseGameFeature::UserProfile::GetDatabasePath() const
```

get path to world database

Returns the path to the current world database.

```cpp
Util::String BaseGameFeature::UserProfile::GetSaveGamePath(const Util::String& saveGameName) const
```

get path to a complete savegame

Get the complete filename to a savegame file.

```cpp
bool BaseGameFeature::UserProfile::HasAttr(const Util::String& name) const
```
return true if attribute exists in the profile

return true if attribute exists in the profile

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Characters::Character Class Reference

#include <charactervariationlibrary.h>

Inheritance diagram for Characters::Character:
Detailed Description

A **Character** resource object holds all shared data belonging to a skinned character.

(C) 2008 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~Character ()</td>
<td>destructor</td>
</tr>
<tr>
<td>Setup ()</td>
<td>setup the character object</td>
</tr>
<tr>
<td>Discard ()</td>
<td>discard the character object</td>
</tr>
<tr>
<td>IsValid () const</td>
<td>return true if character object has been setup</td>
</tr>
<tr>
<td>CharacterSkeleton &amp; Skeleton ()</td>
<td>access to the character's skeleton</td>
</tr>
<tr>
<td>CharacterSkinLibrary &amp; SkinLibrary ()</td>
<td>access to the character's skin library</td>
</tr>
<tr>
<td>CharacterAnimationLibrary &amp; AnimationLibrary ()</td>
<td>access to the character's anim library</td>
</tr>
<tr>
<td>CharacterVariationLibrary &amp; VariationLibrary ()</td>
<td>access to the character's variation library</td>
</tr>
<tr>
<td>GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>IsA (const Rtti &amp;rtti) const</td>
<td></td>
</tr>
</tbody>
</table>
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
</table>

_dumps refcounting leaks, call at end of application (NEBUA3_DEBUG builds only!)_
Member Function Documentation

void
Characters::Character::Setup() 

setup the character object

NOTE: embedded objects must still be setup individually after this
method has been called!

int
Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
Characters::CharacterAnimationController
Characters::CharacterAnimationController
Class Reference

#include <characteranimationcontroller.h>
Detailed Description

Allows to control the animation of a character instance, this is basically a convenience wrapper around Animation::AnimSequencer.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CharacterAnimationController ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~CharacterAnimationController ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Setup</strong> (const Ptr&lt; Jobs::JobPort &gt; &amp;jobPort, const CharacterAnimationLibrary &amp;animLib)</td>
<td>setup the anim controller object</td>
</tr>
<tr>
<td><strong>Discard</strong> ()</td>
<td>discard the anim controller object</td>
</tr>
<tr>
<td><strong>IsValid</strong> () const</td>
<td>return true if object has been setup</td>
</tr>
<tr>
<td><strong>SetupAnimDrivenMotion</strong> (IndexT animDrivenMotionJointIndex)</td>
<td>setup the character instance for anim driven motion</td>
</tr>
<tr>
<td><strong>IsAnimDrivenMotionEnabled</strong> () const</td>
<td>return true if anim driven motion is enabled on the character instance</td>
</tr>
<tr>
<td><strong>GetAnimDrivenMotionVector ()</strong></td>
<td>get computed animation driven motion vector</td>
</tr>
<tr>
<td><strong>EnqueueAnimJob</strong> (const Ptr&lt; Animation::AnimJob &gt; &amp;animJob)</td>
<td>enqueue an anim job</td>
</tr>
<tr>
<td><strong>StopTrack</strong> (IndexT trackIndex, bool allowFadeOut=true)</td>
<td>stop all anim jobs on a given track</td>
</tr>
<tr>
<td><strong>StopAllTracks</strong> (bool allowFadeOut=true)</td>
<td>stop all anim jobs</td>
</tr>
<tr>
<td><strong>AnimSequencer &amp;</strong> Animation::AnimSequencer</td>
<td>direct access to embedded anim sequencer object</td>
</tr>
</tbody>
</table>
const Math::vector &
Characters::CharacterAnimationController::GetAnimDrivenMotionVector( ) [inline]

get computed animation driven motion vector

NOTE: caller needs to make sure that the animation evaluation jobs
have finished!
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Characters::CharacterInstance
Characters::CharacterInstance Class Reference

#include <characterinstance.h>

Inheritance diagram for Characters::CharacterInstance:

```
  Core::RefPtred
    
    Characters::CharacterInstance
```
Detailed Description

Contains the per-instance data of a character.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CharacterInstance ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~CharacterInstance ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Setup</strong> (const Ptr&lt; Character &gt; &amp;origCharacter, const Ptr&lt; Models::ModellInstance &gt; &amp;modelInst)</td>
<td>setup the character instance from a character</td>
</tr>
<tr>
<td><strong>Discard</strong> ()</td>
<td>discard the character instance</td>
</tr>
<tr>
<td><strong>IsValid</strong> () const</td>
<td>return true if the object has been setup</td>
</tr>
<tr>
<td><strong>GetJobPort</strong> () const</td>
<td>get the character instance's job port</td>
</tr>
<tr>
<td><strong>Skeleton</strong> ()</td>
<td>access to get the character's skeleton</td>
</tr>
<tr>
<td><strong>SkinSet</strong> ()</td>
<td>access to character's skin set</td>
</tr>
<tr>
<td><strong>AnimController</strong> ()</td>
<td>access to character's animation controller</td>
</tr>
<tr>
<td><strong>SetVariationSetName</strong> (const Util::StringAtom &amp;variationSetName)</td>
<td>access to the character's variation</td>
</tr>
<tr>
<td><strong>RenderDebug</strong> (const Math::matrix44 &amp;modelTransform)</td>
<td>render a debug visualization of the character</td>
</tr>
<tr>
<td><strong>WaitUpdateDone</strong> () const</td>
<td>wait for the character to become valid after StartUpdateAsync()</td>
</tr>
<tr>
<td><strong>CheckUpdateDone</strong> () const</td>
<td>check whether asynchronous character update has finished</td>
</tr>
<tr>
<td><strong>IsValidForRendering</strong> () const</td>
<td>return whether this object is valid for rendering</td>
</tr>
<tr>
<td>Member</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetJointTextureRowIndex()</code> const</td>
<td>Get character joint texture row index (for GPUTextureSkinning)</td>
</tr>
<tr>
<td><code>GetRefCount()</code> const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code> const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti)</code> const</td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName)</code> const</td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName()</code> const</td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC()</code> const</td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
<td></td>
</tr>
</tbody>
</table>
**Member Function Documentation**

void Characters::CharacterInstance::RenderDebug (const Math::matrix44 modelTransform &)

render a debug visualization of the character

Render a debug visualization of the character.

void Characters::CharacterInstance::WaitUpdateDone ( ) const

wait for the character to become valid after StartUpdateAsync()

Wait until StartUpdate() is done for this character.

bool Characters::CharacterInstance::CheckUpdateDone ( ) const

check whether asynchronous character update has finished

Check if StartUpdate() has finished, return immediately.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Characters::CharacterJoint
#include <characterjoint.h>
Detailed Description

Holds shared data of a character joint.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CharacterJoint()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~CharacterJoint()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Setup(IndexT parentJointIndex, const CharacterJoint* parentJoint, const Math::vector &amp;poseTranslation, const Math::quaternion &amp;poseRotation, const Math::vector &amp;poseScale, const Util::StringAtom &amp;name, Math::matrix44 *invPoseMatrixPtr)</code></td>
<td>Setup the joint</td>
</tr>
<tr>
<td><code>Util::StringAtom &amp; GetName()</code></td>
<td>Get the joint's name</td>
</tr>
<tr>
<td><code>bool HasParentJoint()</code></td>
<td>Return true if the joint has a parent joint</td>
</tr>
<tr>
<td><code>IndexT GetParentJointIndex()</code></td>
<td>Get the parent joint index</td>
</tr>
<tr>
<td><code>const CharacterJoint* GetParentJoint()</code></td>
<td>Get pointer to parent joint (can be 0!)</td>
</tr>
<tr>
<td><code>const Math::vector &amp; GetPoseTranslation()</code></td>
<td>Get pose translation</td>
</tr>
<tr>
<td><code>const Math::quaternion &amp; GetPoseRotation()</code></td>
<td>Get pose rotation</td>
</tr>
<tr>
<td><code>const Math::vector &amp; GetPoseScale()</code></td>
<td>Get pose scale</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetPoseMatrix()</code></td>
<td>Get the pose matrix</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetInvPoseMatrix()</code></td>
<td>Get the inverse pose matrix</td>
</tr>
</tbody>
</table>
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**Characters::CharacterNode**
Characters::CharacterNode Class Reference

#include <characternode.h>

Inheritance diagram for Characters::CharacterNode:
Detailed Description

The `CharacterNode` class wraps a `Character` object into a `ModelNode` for rendering.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CharacterNode()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~CharacterNode()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>OnAttachToModel(const Models::Model &amp;model)</code></td>
<td>called when attached to model node</td>
</tr>
<tr>
<td><code>OnRemoveFromModel()</code></td>
<td>called when removed from model node</td>
</tr>
<tr>
<td><code>LoadResources()</code></td>
<td>called when resources should be loaded</td>
</tr>
<tr>
<td><code>UnloadResources()</code></td>
<td>called when resources should be unloaded</td>
</tr>
<tr>
<td><code>GetResourceState()</code></td>
<td>get overall state of contained resources (Initial, Loaded, Pending, Failed, Cancelled)</td>
</tr>
<tr>
<td><code>OnResourcesLoaded()</code></td>
<td>called once when all pending resource have been loaded</td>
</tr>
<tr>
<td><code>ParseDataTag(const Util::FourCC &amp;fourCC, const IO::BinaryReader &amp;reader)</code></td>
<td>parse data tag (called by loader code)</td>
</tr>
<tr>
<td><code>CreateNodeInstance()</code></td>
<td>create a model node instance</td>
</tr>
<tr>
<td><code>GetCharacter()</code></td>
<td>get the owned character object</td>
</tr>
<tr>
<td><code>GetManagedAnimResource()</code></td>
<td>get the character's managed animation</td>
</tr>
<tr>
<td><code>SetAnimationResourceId(const Resources::ResourceId &amp;resId)</code></td>
<td>set the character's animation resource</td>
</tr>
<tr>
<td><code>GetAnimationResourceId()</code></td>
<td>get the character's animation resource</td>
</tr>
</tbody>
</table>
void 

**SetPosition** (const Math::point &)

set position

const Math::point &

**GetPosition** () const

get position

void 

**SetRotation** (const Math::quaternion &r)

set rotate quaternion

const Math::quaternion &

**GetRotation** () const

get rotate quaternion

void 

**setScale** (const Math::vector &)

set scale

const Math::vector &

**GetScale** () const

get scale

void 

**SetRotatePivot** (const Math::point &p)

set rotate pivot

const Math::point &

**GetRotatePivot** () const

get rotate pivot

void 

**setScalePivot** (const Math::point &p)

set scale pivot

const Math::point &

**GetScalePivot** () const

get scale pivot

bool 

**IsInViewSpace** () const

is transformnode in viewspace

void 

**SetInViewSpace** (bool b)

set transformnode in viewspace

float 

**GetMinDistance** () const

get MinDistance

void 

**SetMinDistance** (float val)

set MinDistance

float 

**GetMaxDistance** () const

get MaxDistance

void 

**SetMaxDistance** (float val)

set MaxDistance

bool 

**LodDistancesUsed** () const

are lod distances used

bool
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetLockedToViewer()</code></td>
<td><code>Get LockedToViewer</code></td>
</tr>
<tr>
<td><code>SetLockedToViewer(bool val)</code></td>
<td><code>set LockedToViewer</code></td>
</tr>
<tr>
<td><code>CheckLodDistance(float distToViewer) const</code></td>
<td><code>helper method to check whether the distance is within lod distances</code></td>
</tr>
<tr>
<td><code>Ptr&lt;ModelNodeInstance&gt;</code></td>
<td><code>CreateNodeInstanceHierarchy()</code></td>
</tr>
<tr>
<td></td>
<td><code>Ptr&lt;ModelInstance&gt; &amp;modelInst)</code></td>
</tr>
<tr>
<td></td>
<td><code>recursively create model node instance model node instances</code></td>
</tr>
<tr>
<td><code>virtual void BeginParseDataTags()</code></td>
<td><code>begin parsing data tags</code></td>
</tr>
<tr>
<td><code>virtual void EndParseDataTags()</code></td>
<td><code>finish parsing data tags</code></td>
</tr>
<tr>
<td><code>virtual void ApplySharedState(IndexT frameIndex)</code></td>
<td><code>apply state shared by all my ModelNodeInstances</code></td>
</tr>
<tr>
<td><code>bool IsAttachedToModel()</code> const</td>
<td><code>return true if currently attached to a Model</code></td>
</tr>
<tr>
<td><code>const Ptr&lt;Model&gt; &amp; GetModel()</code> const</td>
<td><code>get model this node is attached to</code></td>
</tr>
<tr>
<td><code>void SetResourceStreamingLevelOfDetail(float factor)</code></td>
<td><code>sets the resourceStreamingLevelOfDetail but only if the given value is bigger than the current one (reseted on frame-start)</code></td>
</tr>
<tr>
<td><code>void ResetScreenSpaceStats()</code></td>
<td><code>resets all screen space stats e.g. size</code></td>
</tr>
<tr>
<td><code>void SetBoundingBox(const Math::bbox &amp;b)</code></td>
<td><code>set bounding box</code></td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetBoundingBox()</code> const</td>
<td><code>get bounding box of model node</code></td>
</tr>
<tr>
<td><code>void SetName(const Util::StringAtom &amp;)</code></td>
<td><code>set model node name</code></td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetName()</code> const</td>
<td><code>get model node name</code></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void SetType (ModelNodeType::Code)</code></td>
<td>set <code>ModelNodeType</code></td>
</tr>
<tr>
<td><code>ModelNodeType::Code GetType () const</code></td>
<td>get the <code>ModelNodeType</code></td>
</tr>
<tr>
<td><code>void SetParent (const Ptr&lt; ModelNode &gt;&amp;)</code></td>
<td>set parent node</td>
</tr>
<tr>
<td><code>const Ptr&lt; ModelNode &gt;&amp; GetParent () const</code></td>
<td>get parent node</td>
</tr>
<tr>
<td><code>bool HasParent () const</code></td>
<td>return true if node has a parent</td>
</tr>
<tr>
<td><code>void AddChild (const Ptr&lt; ModelNode &gt;&amp;)</code></td>
<td>add a child node</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Ptr&lt; ModelNode &gt; &gt;&amp; GetChildren () const</code></td>
<td>get child nodes</td>
</tr>
<tr>
<td><code>bool HasChild (const Util::StringAtom&amp; name)</code></td>
<td>return true if a direct child exists by name</td>
</tr>
<tr>
<td><code>const Ptr&lt; ModelNode &gt;&amp; LookupChild (const Util::StringAtom&amp; name)</code></td>
<td>get pointer to direct child by name</td>
</tr>
<tr>
<td><code>void AddVisibleNodeInstance (IndexT frameIndex, const Ptr&lt; ModelNodeInstance &gt;&amp; node)</code></td>
<td>called by model node instance on Notify</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Ptr&lt; ModelNodeInstance &gt; &gt;&amp; GetVisibleModelNodeInstances (ModelNodeType::Code t)</code></td>
<td>get visible model node instances</td>
</tr>
<tr>
<td><code>bool HasStringAttr (const Util::StringAtom&amp; attrId)</code></td>
<td>has string attr</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetStringAttr (const Util::StringAtom&amp; attrId) const</code></td>
<td>get string attr</td>
</tr>
<tr>
<td><code>void SetStringAttr (const Util::StringAtom&amp; attrId)</code></td>
<td>set string attr</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>add string attribute</td>
<td>void &amp;attrId, const Util::StringAtom &amp;value)</td>
</tr>
<tr>
<td>get the current refcount</td>
<td>int GetRefCount () const</td>
</tr>
<tr>
<td>increment refcount by one</td>
<td>void AddRef ()</td>
</tr>
<tr>
<td>decrement refcount and destroy object if refcount is zero</td>
<td>void Release ()</td>
</tr>
<tr>
<td>return true if this object is instance of given class</td>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class by string</td>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class by fourcc</td>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>return true if this object is instance of a derived class</td>
<td>bool IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>return true if this object is instance of a derived class by string</td>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td>return true if this object is instance of a derived class by fourcc</td>
<td>bool IsA (const Util::FourCC &amp;rtti) const</td>
</tr>
<tr>
<td>get the class name</td>
<td>const Util::String &amp; GetClassName () const</td>
</tr>
<tr>
<td>get the class FourCC code</td>
<td>Util::FourCC GetClassFourCC () const</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>resourceStreamingLevelOfDetail</code></td>
<td>factor between 0.0 (close) and 1.0 (far away) describing the distance to camera (used for decision of max needed mipMap)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Characters::CharacterNode::OnResourcesLoaded() [virtual]
called once when all pending resource have been loaded
Called when all resources of this Model are loaded. We need to setup the animation and variation libraries once this has happened.
Reimplemented from Models::ModelNode.

Ptr&lt; ModelNodeInstance &gt;
Models::ModelNode::CreateNodeInstanceHierarchy ( const Ptr&lt; ModelInstance &gt; modellInst ) [inherited]
recursively create model node instance and child model node instances
Create the node instance hierarchy.

void
Models::ModelNode::BeginParseDataTags() [virtual, inherited]
begin parsing data tags
Begin parsing data tags. This method is called by StreamModelLoader before ParseDataTag() is called for the first time.

void
Models::ModelNode::EndParseDataTags() [virtual, inherited]
finish parsing data tags
End parsing data tags. This method is called by StreamModelLoader after the last ParseDataTag() is called.

void
Models::ModelNode::ApplySharedState ( IndexT frameIndex ) [virtual, inherited]
apply state shared by all my ModelNodeInstances

This method is called once before rendering the ModelNode's visible instance nodes through the ModelNodeInstance::ApplyState() and ModelNodeInstance::Render() methods. The method must apply the shader state that is shared across all instances. Since this state is constant across all instance nodes, this only happens once before rendering an instance set.

Reimplemented in Characters::CharacterSkinNode, Models::ShapeNode, and Models::StateNode.

```cpp
void Models::ModelNode::ResetScreenSpaceStats() [inline, inherited]
```
resets all screen space stats e.g. size

Reset resourceStreamingLevelOfDetail to -1.0 as we are able to recognize invisible items this way. (visible items will overwrite this value with a value >= 0.0)

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const 

Core::RefCounted::GetClassName () const [inline, inherited]

get the class name

Get the class name of the object.

Core::RefCounted::GetClassFourCC () const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void

Core::RefCounted::DumpRefCountingLeaks () [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Characters::CharacterNodeInstance
#include <characternodeinstance.h>

Inheritance diagram for Characters::CharacterNodeInstance:

```
Core::RefCounted
  ↓
Models::ModelNodeInstance
  ↓
Models::TransformNodeInstance
  ↓
Characters::CharacterNodeInstance
```
Detailed Description

The **CharacterNodeInstance** class wraps a **CharacterInstance** object into a ModelNodeInstance class.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CharacterNodeInstance()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>virtual ~CharacterNodeInstance()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual void OnNotifyCullingVisible(IndexT frameIndex, Timing::Time time)</code></td>
<td>Called from <code>ModelEntity::OnNotifyCullingVisible</code></td>
</tr>
<tr>
<td><code>const Ptr&lt; CharacterInstance &gt; &amp; GetCharacterInstance()</code></td>
<td>Get the node's character instance object</td>
</tr>
<tr>
<td><code>virtual void OnRenderBefore(IndexT frameIndex, Timing::Time time)</code></td>
<td>Called from <code>ModelEntity::OnRenderBefore</code></td>
</tr>
<tr>
<td><code>virtual void ApplyState()</code></td>
<td>Apply per-instance state prior to rendering</td>
</tr>
<tr>
<td><code>void SetPosition(const Math::point &amp;p)</code></td>
<td>Set position</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetPosition()</code></td>
<td>Get position</td>
</tr>
<tr>
<td><code>void SetRotate(const Math::quaternion &amp;r)</code></td>
<td>Set rotate quaternion</td>
</tr>
<tr>
<td><code>const Math::quaternion &amp; GetRotate()</code></td>
<td>Get rotate quaternion</td>
</tr>
<tr>
<td><code>void SetScale(const Math::vector &amp;s)</code></td>
<td>Set scale</td>
</tr>
<tr>
<td><code>const Math::vector &amp; GetScale()</code></td>
<td>Get scale</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void SetRotatePivot (const Math::point &amp;p)</code></td>
<td>set rotate pivot</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetRotatePivot () const</code></td>
<td>get rotate pivot</td>
</tr>
<tr>
<td><code>void SetScalePivot (const Math::point &amp;p)</code></td>
<td>set scale pivot</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetScalePivot () const</code></td>
<td>get scale pivot</td>
</tr>
<tr>
<td><code>void SetOffsetMatrix (const Math::matrix44 &amp;m)</code></td>
<td>set optional offset matrix</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetOffsetMatrix () const</code></td>
<td>get optional offset matrix</td>
</tr>
<tr>
<td><code>bool IsInViewSpace () const</code></td>
<td>is transformnode in viewspace</td>
</tr>
<tr>
<td><code>void SetInViewSpace (bool b)</code></td>
<td>set transformnode in viewspace</td>
</tr>
<tr>
<td><code>bool GetLockedToViewer () const</code></td>
<td>get LockedToViewer</td>
</tr>
<tr>
<td><code>void SetLockedToViewer (bool val)</code></td>
<td>set LockedToViewer</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetLocalTransform ()</code></td>
<td>get resulting local transform matrix in local parent space</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetModelTransform () const</code></td>
<td>get model space transform (valid after Update())</td>
</tr>
<tr>
<td><code>virtual void DiscardHierarchy ()</code></td>
<td>discard the model node instance and all of its children</td>
</tr>
<tr>
<td><code>bool IsValid () const</code></td>
<td>return true if the model node instance is valid</td>
</tr>
<tr>
<td><code>virtual void OnVisibilityResolve (IndexT resolveIndex, float)</code></td>
<td></td>
</tr>
</tbody>
</table>
virtual void Render ()
perform rendering
called during visibility resolve

const Util::StringAtom & GetName () const
get model node name

bool HasParent () const
return true if node has a parent

const Ptr< ModelNodeInstance > & GetParent () const
get parent node

const Util::Array< Ptr< ModelNodeInstance > > & GetChildren () const
get child nodes

bool HasChild (const Util::StringAtom &name) const
return true if a direct child exists by name

const Ptr< ModelNodeInstance > & LookupChild (const Util::StringAtom &name) const
get pointer to direct child by name

Ptr< ModelNodeInstance > LookupPath (const Util::String &path)
get modelnodeinstance by hierarchy path

const Ptr< ModelInstance > & GetModellInstance () const
get the ModelInstance we are attached to

const Ptr< ModelNode > & GetModelNode () const
get the ModelNode we’re associated with

void SetVisible (bool b, Timing::Time time, bool recursive=true)
set the node instance’s visibility

bool IsVisible () const
return true if node instance is set to visible
<table>
<thead>
<tr>
<th><strong>IndexT</strong></th>
<th><strong>GetModelNodeInstanceIndex</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get model node instance index for current frame</td>
</tr>
<tr>
<td>int</td>
<td><strong>GetRefCount</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td><strong>AddRef</strong> ()</td>
</tr>
<tr>
<td></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void</td>
<td><strong>Release</strong> ()</td>
</tr>
<tr>
<td></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void Setup (const Ptr&lt; Models::ModelInstance &gt; &amp;inst, const Ptr&lt; Models::ModelNode &gt; &amp;node, const Ptr&lt; Models::ModelNodeInstance &gt; &amp;parentNodeInst)</td>
<td>called when attached to ModelInstance</td>
</tr>
<tr>
<td>virtual void Discard ()</td>
<td>called when removed from ModelInstance</td>
</tr>
<tr>
<td>virtual void RenderDebug ()</td>
<td>render node specific debug shape</td>
</tr>
<tr>
<td>void SetParent (const Ptr&lt; ModelNodeInstance &gt; &amp;p)</td>
<td>set parent node</td>
</tr>
<tr>
<td>void AddChild (const Ptr&lt; ModelNodeInstance &gt; &amp;c)</td>
<td>add a child node</td>
</tr>
<tr>
<td>virtual void OnShow (Timing::Time time)</td>
<td>called when the node becomes visible with current time</td>
</tr>
<tr>
<td>virtual void OnHide (Timing::Time time)</td>
<td>called when the node becomes invisible</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::TransformNodeInstance::OnRenderBefore ( IndexT frameIndex, Timing::Time time ) [virtual, inherited]
```

called from ModelEntity::OnRenderBefore

The update method should first invoke any animators which change per-instance attributes (this is done in the parent class). Then the local space transforms must be flattened into model space.

**NOTE:** this method must be called late in the frame to give other systems a chance to modify the transform matrix (for instance the character attachment system).

Reimplemented from `Models::ModelNodeInstance`.

Reimplemented in `Particles::ParticleSystemNodeInstance`.

```cpp
void Models::TransformNodeInstance::ApplyState ( ) [virtual, inherited]
```

apply per-instance state prior to rendering

Set our model matrix (computed in the Update() method) as current model matrix in the TransformDevice.

Reimplemented from `Models::ModelNodeInstance`.

Reimplemented in `Models::StateNodeInstance`, and `Particles::ParticleSystemNodeInstance`.

```cpp
void Models::ModelNodeInstance::DiscardHierarchy ( ) [virtual, inherited]
```

discard the model node instance and all of its children
Discards this model node instance and all of its children recursively.

```cpp
void Models::ModelNodeInstance::Render() [virtual, inherited]
```

perform rendering

The `Render()` method is called when the `ModelNodeInstance` needs to render itself. There will always be a call to the `Apply()` method before `Render()` is called, however, `Render()` may be called several times per `Apply()` invocation.

Reimplemented in `Characters::CharacterSkinNodeInstance`, `Models::ShapeNodeInstance`, and `Particles::ParticleSystemNodeInstance`.

```cpp
bool Models::ModelNodeInstance::IsVisible() const [inherited]
```

return true if node instance is set to visible

FIXME: The recursion in this is method makes it slow, especially in deep hierarchies. Also it can't be inlined.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:41 2010
Characters::CharacterServer
Characters::CharacterServer Class Reference

#include <characterserver.h>

Inheritance diagram for Characters::CharacterServer:
Detailed Description

Handles central aspects of the character system.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CharacterServer()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~CharacterServer()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Setup()</strong></td>
<td>setup the character server</td>
</tr>
<tr>
<td><strong>void Discard()</strong></td>
<td>discard the character server</td>
</tr>
<tr>
<td><strong>bool IsValid() const</strong></td>
<td>return true if character server has been setup</td>
</tr>
<tr>
<td><strong>getSkinningTechnique() const</strong></td>
<td>get the skinning technique used by this platform</td>
</tr>
<tr>
<td><strong>void BeginFrame(IndexT frameIndex)</strong></td>
<td>begin frame</td>
</tr>
<tr>
<td><strong>void BeginGather()</strong></td>
<td>begin gathering phase (there may be several per frame)</td>
</tr>
<tr>
<td><strong>void GatherVisibleCharacter(const Ptr&lt;CharacterInstance&gt; &amp;charInst, Timing::Time time)</strong></td>
<td>register a visible character (same character may be registered several times per frame!)</td>
</tr>
<tr>
<td><strong>void GatherSkinMesh(const Ptr&lt;CharacterInstance&gt; &amp;charInst, const Ptr<a href="">CoreGraphics::Mesh</a> &amp;srcMesh)</strong></td>
<td>update a character skin</td>
</tr>
<tr>
<td><strong>void EndGather()</strong></td>
<td>finish gathering phase</td>
</tr>
<tr>
<td><strong>void StartUpdateCharacterSkeletons()</strong></td>
<td>update visible characters (update animation, evaluate skeleton, potentially perform software skinning)</td>
</tr>
<tr>
<td><strong>void UpdateCharacterSkins()</strong></td>
<td>update character skins (if software skinning)</td>
</tr>
</tbody>
</table>
void BeginDraw ()

begin render phase

void DrawSoftwareSkinnedMesh (SkinnedMeshRenderer::DrawHandle h, IndexT primGroupIndex)

draw a software skinned mesh

void DrawGPUSkinnedMesh (const Ptr<Characters::CharacterInstance > &charInst, const Ptr<CoreGraphics::Mesh > &mesh, IndexT primGroupIndex, const Util::Array< IndexT > &jointPalette, const Ptr<CoreGraphics::ShaderVariable > &jointPaletteShdVar)

draw a hardware skinned mesh

void DrawGPUTextureSkinnedMesh (const Ptr<Characters::CharacterInstance > &charInst, const Ptr<CoreGraphics::Mesh > &mesh, IndexT primGroupIndex, const Ptr<CoreGraphics::ShaderVariable > &charInstShaderVar)

draw a skinned mesh

void EndDraw ()

end render phase

void EndFrame ()

end current frame

int GetRefCount () const

get the current refcount

void AddRef ()

increment refcount by one

void Release ()

decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const

return true if this object is instance of given class

bool IsInstanceOf (const Util::String
<table>
<thead>
<tr>
<th>bool</th>
<th>&amp;className) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><code>DumpRefCountingLeaks ()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Characters::CharacterServer::StartUpdateCharacterSkeletons()
update visible characters (update animation, evaluate skeleton, potentially perform software skinning)
Start updating character skeletons. This is an asynchronous operation!

int
Core::RefCounted::GetRefCount()
const [inline, inherited]
get the current refcount
Return the current refcount of the object.

void
Core::RefCounted::AddRef()
[inline, inherited]
increment refcount by one
Increment the refcount of the object.

void
Core::RefCounted::Release()
[inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName()
const [inline, inherited]
get the class name
Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC()
const [inline, inherited]
Get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Characters::CharacterSkeleton
Characters::CharacterSkeleton Class Reference

#include <characterskeleton.h>
Detailed Description

Contains the skeleton data of a character which is shared between all instances of the character.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CharacterSkeleton ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~CharacterSkeleton ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Setup (SizeT numJoints)</strong></td>
<td>setup the skeleton</td>
</tr>
<tr>
<td><strong>void SetupJoint (IndexT jointIndex, IndexT parentJointIndex, const Math::point &amp;poseTranslation, const Math::quaternion &amp;poseRotation, const Math::vector &amp;poseScale, const Util::StringAtom &amp;name)</strong></td>
<td>setup a joint in the skeleton</td>
</tr>
<tr>
<td><strong>void Discard ()</strong></td>
<td>discard the skeleton</td>
</tr>
<tr>
<td><strong>bool IsValid () const</strong></td>
<td>return true if object has been setup</td>
</tr>
<tr>
<td><strong>SizeT GetNumJoints () const</strong></td>
<td>get number of joints in the skeleton</td>
</tr>
<tr>
<td><strong>const CharacterJoint &amp; GetJoint (IndexT index) const</strong></td>
<td>access to a joint in the skeleton</td>
</tr>
<tr>
<td><strong>IndexT GetJointIndexByName (const Util::StringAtom &amp;jointName)</strong></td>
<td>get a joint index by name, returns InvalidIndex, if joint not found</td>
</tr>
<tr>
<td><strong>const Util::FixedArray&lt; Math::matrix44 &gt; &amp; GetInvPoseMatrixArray () const</strong></td>
<td>get the invPoseMatrixArray (contains inverse joint pose matrices)</td>
</tr>
<tr>
<td><strong>const Util::FixedArray&lt; Math::float4 &gt; &amp; GetDefaultSamplesArray () const</strong></td>
<td>get pointer to default samples if no valid anim is set on character</td>
</tr>
</tbody>
</table>
Member Function Documentation

const Util::FixedArray< Math::float4 > & Characters::CharacterSkeleton::GetDefaultSamplesArray() const [inline]

get pointer to default samples if no valid anim is set on character

Get pointer to the default samples array. This is used if no animation is set on the character as input to the skeleton evaluation. Format is always translation, rotation, scale, velocity.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Characters::CharacterSkeletonInstance
Characters::CharacterSkeletonInstance
Class Reference

#include <characterskeletoninstance.h>
Detailed Description

Contains the per-instance skeleton data of a character.

(C) 2008 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CharacterSkeletonInstance()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~CharacterSkeletonInstance()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Setup(const CharacterSkeleton &amp;skeleton)</code></td>
<td>Setup from <code>CharacterSkeleton</code></td>
</tr>
<tr>
<td><code>Discard()</code></td>
<td>Discard the object</td>
</tr>
<tr>
<td><code>IsValid()</code> const</td>
<td>Return true if the object has been setup</td>
</tr>
<tr>
<td><code>RenderDebug(const Math::matrix44 &amp;modelTransform)</code></td>
<td>Render a debug visualization of the character</td>
</tr>
<tr>
<td><code>GetNumJoints()</code> const</td>
<td>Get number of joint instances in the skeleton</td>
</tr>
<tr>
<td><code>GetSkinMatrixArray()</code> const</td>
<td>Get the skin matrix array</td>
</tr>
<tr>
<td><code>GetJointMatrix(IndexT i)</code> const</td>
<td>Get a joint matrix by joint index</td>
</tr>
<tr>
<td><code>ApplyJointComponents(const Util::FixedArray&lt;CharJointComponents&gt; &amp;set)</code></td>
<td>Apply joint components</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void Characters::CharacterSkeletonInstance::Setup(
    const CharacterSkeleton& skeleton)
```

setup from **CharacterSkeleton**

NOTE: sampleBuffer is a pointer to animation samples which are the result of the character's animation.

```cpp
void Characters::CharacterSkeletonInstance::RenderDebug(
    const Math::matrix44& modelTransform)
```

render a debug visualization of the character

Render a debug visualization of the character. Note: The debug visualization isn't up-to-date, one frame after skinning cause RenderDebug is called after view->Render where all debug shapes are rendered (see framebatch)
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Characters::CharacterSkin
Characters::CharacterSkin Class Reference

#include <characterskin.h>
Detailed Description

Describes a single skin in a CharacterSkinLibrary.

(C) 2008 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharacterSkin ()</td>
<td>default constructor</td>
</tr>
</tbody>
</table>
| CharacterSkin (const Ptr<
  Models::ModelNode > &modelNode,
  const Util::StringAtom &category,
  const Util::StringAtom &name) | constructor                                                                  |
| ~CharacterSkin ()                | destructor                                                                   |
| const Ptr< Models::ModelNode > & | GetModelNode () const                                                       |
|                                  | get pointer to skin's model node                                            |
| const Util::StringAtom &         | GetName () const                                                            |
|                                  | get skin name                                                                |
| const Util::StringAtom &         | GetCategory () const                                                         |
|                                  | get skin category                                                            |

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:41 2010
Characters::CharacterSkinLibrary
Characters::CharacterSkinLibrary
Class Reference

#include <characterskinlibrary.h>
Detailed Description

Library of all available skins of a character.

(C) 2008 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharacterSkinLibrary ()</td>
<td>constructor</td>
</tr>
<tr>
<td>~CharacterSkinLibrary ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void ReserveSkins (SizeT numSkins)</td>
<td>reserve skins (if number of skins is known beforehand)</td>
</tr>
<tr>
<td>void AddSkin (const CharacterSkin &amp;skin)</td>
<td>add a skin</td>
</tr>
<tr>
<td>void ReserveSkinLists (SizeT numSkinLists)</td>
<td>reserve skin lists (if number of skins is known beforehand)</td>
</tr>
<tr>
<td>void AddSkinList (const CharacterSkinList &amp;skinList)</td>
<td>add a skin list</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>discard object (clears skins and skin lists)</td>
</tr>
<tr>
<td>SizeT GetNumSkins () const</td>
<td>get number of skins in the library</td>
</tr>
<tr>
<td>const CharacterSkin &amp; GetSkin (IndexT i) const</td>
<td>get skin at index</td>
</tr>
<tr>
<td>bool HasSkin (const Util::StringAtom &amp;name) const</td>
<td>return true if skin exists by name</td>
</tr>
<tr>
<td>const CharacterSkin &amp; GetSkinByName (const Util::StringAtom &amp;name) const</td>
<td>get skin by name</td>
</tr>
<tr>
<td>IndexT GetSkinIndexByName (const Util::StringAtom &amp;name) const</td>
<td>get skin index by name</td>
</tr>
<tr>
<td>SizeT GetNumSkinLists () const</td>
<td>get number of skin lists in the library</td>
</tr>
<tr>
<td>const CharacterSkinList &amp; GetSkinList (IndexT i) const</td>
<td>get skin list at index</td>
</tr>
<tr>
<td>HasSkinList (const Util::StringAtom &amp;name)</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>const</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>return true if skin list exists by name</td>
</tr>
<tr>
<td>const CharacterSkinList &amp;</td>
<td>GetSkinListByName (const Util::StringAtom &amp;name) const</td>
</tr>
<tr>
<td>get skin list by name</td>
<td></td>
</tr>
<tr>
<td>IndexT</td>
<td>GetSkinListIndexByName (const Util::StringAtom &amp;name) const</td>
</tr>
<tr>
<td>get skin list index by name</td>
<td></td>
</tr>
</tbody>
</table>
Characters::CharacterSkinList
Characters::CharacterSkinList Class Reference

#include <characterskinlist.h>
Detailed Description

A skin list contains a set of visible skins for a character.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharacterSkinList ()</td>
<td><code>default constructor</code></td>
</tr>
<tr>
<td>~CharacterSkinList ()</td>
<td><code>destructor</code></td>
</tr>
<tr>
<td>void SetName (const Util::StringAtom &amp;name)</td>
<td>set the skin list's name</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; GetName () const</td>
<td>get the skin list's name</td>
</tr>
<tr>
<td>void SetSkins (const Util::Array<a href="">Util::StringAtom</a> &amp;skins)</td>
<td>set the skins in the skin list</td>
</tr>
<tr>
<td>const Util::Array<a href="">Util::StringAtom</a> &amp; GetSkins () const</td>
<td>get the skins</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Characters::CharacterSkinNode
Characters::CharacterSkinNode Class Reference

#include <characterskinnode.h>

Inheritance diagram for Characters::CharacterSkinNode:
Detailed Description

A ModelNode which wraps a character skin mesh.

(C) 2008 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CharacterSkinNode()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~CharacterSkinNode()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual Ptr&lt; Models::ModelNodeInstance &gt;</td>
<td>CreateNodeInstance() const create a model node instance</td>
</tr>
<tr>
<td>virtual bool ParseDataTag(const Util::FourCC &amp;fourCC, const Ptr&lt; IO::BinaryReader &gt; &amp;reader)</td>
<td>parse data tag (called by loader code)</td>
</tr>
<tr>
<td>virtual void ApplySharedState(IndexT frameIndex)</td>
<td>apply state shared by all my ModelNodeInstances</td>
</tr>
<tr>
<td>void ReserveFragments(SizeT numFragments)</td>
<td>reserve fragments (call before adding fragments)</td>
</tr>
<tr>
<td>void AddFragment(IndexT primGroupIndex, const Util::Array&lt;IndexT&gt; &amp;jointPalette)</td>
<td>add a fragment (consisting of a mesh group and a joint palette)</td>
</tr>
<tr>
<td>SizeT GetNumFragments() const</td>
<td>get number of skin fragments</td>
</tr>
<tr>
<td>IndexT GetFragmentPrimGroupIndex(IndexT fragmentIndex) const</td>
<td>get primitive group index of a fragment</td>
</tr>
<tr>
<td>const Util::Array&lt;IndexT&gt; &amp; GetFragmentJointPalette(IndexT fragmentIndex) const</td>
<td>get joint palette of a fragment</td>
</tr>
<tr>
<td>virtual void LoadResources()</td>
<td>called when resources should be loaded</td>
</tr>
<tr>
<td>virtual void UnloadResources()</td>
<td>called when resources should be unloaded</td>
</tr>
</tbody>
</table>
| virtual Resources::Resource::State GetResourceState() const | }
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetMeshResourceId (const Resources::ResourceId &amp;resId)</code></td>
<td>Set mesh resource id</td>
</tr>
<tr>
<td><code>const Resources::ResourceId &amp; GetMeshResourceId () const</code></td>
<td>Get mesh resource id</td>
</tr>
<tr>
<td><code>void SetPrimitiveGroupIndex (IndexT i)</code></td>
<td>Set primitive group index</td>
</tr>
<tr>
<td><code>IndexT GetPrimitiveGroupIndex () const</code></td>
<td>Get primitive group index</td>
</tr>
<tr>
<td><code>void SetMeshResourceLoader (const Resources::ResourceLoader &amp;loader)</code></td>
<td>Set optional resourceloader</td>
</tr>
<tr>
<td><code>const Ptr&lt; Resources::ManagedMesh &gt; &amp; GetManagedMesh () const</code></td>
<td>Get managed mesh</td>
</tr>
<tr>
<td><code>void SetShader (const Resources::ResourceId &amp;resId)</code></td>
<td>Set shader resource id</td>
</tr>
<tr>
<td><code>const Resources::ResourceId &amp; GetShader () const</code></td>
<td>Get shader resource id</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp; GetShaderInstance () const</code></td>
<td>Get pointer to contained shader instance</td>
</tr>
<tr>
<td><code>void AddShaderParam (const Util::String &amp;paramName, const Util::Variant &amp;paramValue)</code></td>
<td>Add optional shader parameter, must be called before LoadResources</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Util::KeyValuePair&lt;Util::StringAtom, Util::Variant&gt; &gt; &amp; GetShaderParameter () const</code></td>
<td>Get shaderparams</td>
</tr>
<tr>
<td><code>void SetPosition (const Math::point &amp;)</code></td>
<td>Set position</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetPosition () const</code></td>
<td>Get position</td>
</tr>
<tr>
<td><code>void SetRotation (const Math::quaternion &amp;r)</code></td>
<td>Set rotation</td>
</tr>
</tbody>
</table>
const Math::quaternion & GetRotation () const
get rotate quaternion

void SetScale (const Math::vector & s)
set scale

const Math::vector & GetScale () const
get scale

void SetRotatePivot (const Math::point & p)
set rotate pivot

const Math::point & GetRotatePivot () const
get rotate pivot

void SetScalePivot (const Math::point & p)
set scale pivot

const Math::point & GetScalePivot () const
get scale pivot

bool IsInViewSpace () const
is transformnode in viewspace

void SetInViewSpace (bool b)
set transformnode in viewspace

float GetMinDistance () const
get MinDistance

void SetMinDistance (float val)
set MinDistance

float GetMaxDistance () const
get MaxDistance

void SetMaxDistance (float val)
set MaxDistance

bool LodDistancesUsed () const
are lod distances used

virtual void OnAttachToModel (const Ptr<Model> & model)
called when attached to model node

bool GetLockedToViewer () const
get LockedToViewer

void SetLockedToViewer (bool val)
set LockedToViewer
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool CheckLodDistance (float distToViewer) const</td>
<td>helper method to check whether the distance within lod distances</td>
</tr>
<tr>
<td>Ptr&lt; ModelNodeInstance &gt; CreateNodeInstanceHierarchy (Ptr&lt; ModelInstance &gt; &amp;modelInst)</td>
<td>recursively create model node instance and model node instances</td>
</tr>
<tr>
<td>virtual void OnRemoveFromModel ()</td>
<td>called when removed from model node</td>
</tr>
<tr>
<td>virtual void OnResourcesLoaded ()</td>
<td>called once when all pending resource have loaded</td>
</tr>
<tr>
<td>virtual void BeginParseDataTags ()</td>
<td>begin parsing data tags</td>
</tr>
<tr>
<td>virtual void EndParseDataTags ()</td>
<td>finish parsing data tags</td>
</tr>
<tr>
<td>bool IsAttachedToModel () const</td>
<td>return true if currently attached to a Model</td>
</tr>
<tr>
<td>const Ptr&lt; Model &gt; &amp; GetModel () const</td>
<td>get model this node is attached to</td>
</tr>
<tr>
<td>void SetResourceStreamingLevelOfDetail (float factor)</td>
<td>sets the resourceStreamingLevelOfDetail but if the given value is bigger than the current one (reseted on frame-start)</td>
</tr>
<tr>
<td>void ResetScreenSpaceStats ()</td>
<td>resets all screen space stats e.g. size</td>
</tr>
<tr>
<td>void SetBoundingBox (const Math::bbox &amp;b)</td>
<td>set bounding box</td>
</tr>
<tr>
<td>const Math::bbox &amp; GetBoundingBox () const</td>
<td>get bounding box of model node</td>
</tr>
<tr>
<td>void SetName (const Util::StringAtom &amp;)</td>
<td>set model node name</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; GetName () const</td>
<td>get model node name</td>
</tr>
<tr>
<td>void SetType (ModelNodeType::Code)</td>
<td>set model node type</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>ModelNodeType::Code</code></td>
<td>set <code>ModelNodeType</code></td>
</tr>
<tr>
<td><code>GetNodeType</code></td>
<td>get the <code>ModelNodeType</code></td>
</tr>
<tr>
<td><code>SetParent (const Ptr&lt; ModelNode &gt;&amp; p)</code></td>
<td>set parent node</td>
</tr>
<tr>
<td><code>GetParent () const</code></td>
<td>get parent node</td>
</tr>
<tr>
<td><code>HasParent () const</code></td>
<td>return true if node has a parent</td>
</tr>
<tr>
<td><code>AddChild (const Ptr&lt; ModelNode &gt;&amp; c)</code></td>
<td>add a child node</td>
</tr>
<tr>
<td><code>GetChildren () const</code></td>
<td>get child nodes</td>
</tr>
<tr>
<td><code>HasChild (const Util::StringAtom &amp; name)</code></td>
<td>return true if a direct child exists by name</td>
</tr>
<tr>
<td><code>LookupChild (const Util::StringAtom &amp; name)</code></td>
<td>get pointer to direct child by name</td>
</tr>
<tr>
<td><code>AddVisibleNodeInstance (IndexT frameIndex, const Ptr&lt; ModelNodeInstance &gt;&amp; nodeInst)</code></td>
<td>called by model node instance on <code>NotifyVisible()</code></td>
</tr>
<tr>
<td><code>GetVisibleModelNodeInstances (ModelNodeType::Code t)</code></td>
<td>get visible model node instances</td>
</tr>
<tr>
<td><code>HasStringAttr (const Util::StringAtom &amp; attrId)</code></td>
<td>has string attr</td>
</tr>
<tr>
<td><code>GetStringAttr (const Util::StringAtom &amp; attrId)</code></td>
<td>get string attr</td>
</tr>
</tbody>
</table>
| `SetStringAttr (const Util::StringAtom & attrId, const Util::StringAtom & value)` | set string attr
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class or derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class or derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class or derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
</table>

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
## Protected Member Functions

<p>| void         | <strong>SetupManagedTextureVariable</strong> (const Resources::ResourceId &amp;texResId, const Ptr<a href="">CoreGraphics::ShaderVariable</a> &amp;var) | setup a new managed texture variable |
| void         | <strong>UpdateManagedTextureVariables</strong> (IndexT frameIndex) | update managed texture variables |
| virtual Ptr&lt; ModelNodeInstance &gt; | <strong>RecurseCreateNodeInstanceHierarchy</strong> (const Ptr&lt;ModelInstance&gt; &amp;modelInst, const Ptr&lt;ModelNodeInstance&gt; &amp;parentNodeInst) | recursively create node instance hierarchy |</p>
<table>
<thead>
<tr>
<th>float</th>
<th>resourceStreamingLevelOfDetail</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>factor between 0.0 (close) and 1.0 (far away) describing the distance to camera (used for decision of max needed mipMap)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::StateNode::AddShaderParam(
    const Util::String& paramName,
    const Util::Variant& paramValue
) [inherited]
```

add optional shader parameter, must be called before LoadResources

Manual shader parameters must be added before LoadResources is called, because on LoadResources all shaderparams are validated.

```cpp
void Models::StateNode::SetupManagedTextureVariable(
    const Resources::ResourceId& texResId,
    const Ptr<CoreGraphics::ShaderVariable>& var
) [protected, inherited]
```

setup a new managed texture variable

Create a new managed texture resource and bind it to the provided shader variable.

```cpp
void Models::StateNode::UpdateManagedTextureVariables(
    IndexT frameIndex
) [protected, inherited]
```

update managed texture variables

This method transfers texture from our managed texture objects into their associated shader variable. This is necessary since the actual texture of a managed texture may change from frame to frame because of resource management.

```cpp
Ptr<ModelNodeInstance>
Models::ModelNode::CreateNodeInstanceHierarchy(
    const Ptr<ModelInstance>& modellInst
) [inherited]
```
recursively create model node instance and child model node instances

Create the node instance hierarchy.

```cpp
void Models::ModelNode::OnResourcesLoaded() [virtual, inherited]
```
called once when all pending resource have been loaded

This method is called once by Model::OnResourcesLoaded() when all pending resources of a model have been loaded.

Reimplemented in Characters::CharacterNode, and Particles::ParticleSystemNode.

```cpp
void Models::ModelNode::BeginParseDataTags() [virtual, inherited]
```
begin parsing data tags

Begin parsing data tags. This method is called by StreamModelLoader before ParseDataTag() is called for the first time.

```cpp
void Models::ModelNode::EndParseDataTags() [virtual, inherited]
```
finish parsing data tags

End parsing data tags. This method is called by StreamModelLoader after the last ParseDataTag() is called.

```cpp
void Models::ModelNode::ResetScreenSpaceStats() [inline, inherited]
```
resets all screen space stats e.g. size

Reset resourceStreamingLevelOfDetail to -1.0 as we are able to recognize invisible items this way. (visible items will overwrite this value with a value >= 0.0)
`Models::ModelNode::RecurseCreateNodeInstanceHierarchy` recursively create node instance hierarchy

Recursively create node instances and attach them to the provided model instance. Returns a pointer to the root node instance.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]`
get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Characters::CharacterSkinNodeInstance
#include <characterskinnodeinstance.h>

Inheritance diagram for Characters::CharacterSkinNodeInstance:
Detailed Description

Implements per-instance functionality of a character skin node.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CharacterSkinNodeInstance</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~CharacterSkinNodeInstance</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void Setup (const Ptr&lt; Models::ModelInstance &gt;&amp; inst, const Ptr&lt; Models::ModelNode &gt;&amp; node, const Ptr&lt; Models::ModelNodeInstance &gt;&amp; parentNodeInst)</td>
<td>called when attached to ModelInstance</td>
</tr>
<tr>
<td>virtual void Discard ()</td>
<td>called when removed from ModelInstance</td>
</tr>
<tr>
<td>virtual void OnNotifyCullingVisible (IndexT frameIndex, Timing::Time time)</td>
<td>called from ModelEntity::OnNotifyCullingVisible</td>
</tr>
<tr>
<td>virtual void Render ()</td>
<td>perform rendering</td>
</tr>
<tr>
<td>virtual void OnVisibilityResolve (IndexT resolveIndex, float distToViewer)</td>
<td>called during visibility resolve</td>
</tr>
<tr>
<td>virtual void ApplyState ()</td>
<td>apply per-instance state prior to rendering</td>
</tr>
<tr>
<td><strong>Ptr&lt; CoreGraphics::ShaderVariableInstance &gt;</strong></td>
<td>CreateShaderVariableInstance (CoreGraphics::ShaderVariableInstance semantic)</td>
</tr>
<tr>
<td></td>
<td>instanciate a shader variable by semantic</td>
</tr>
<tr>
<td>bool HasShaderVariableInstance (CoreGraphics::ShaderVariableInstance semantic) const</td>
<td>return true if a shader variable has been instantiated</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; CoreGraphics::ShaderVariableInstance &gt; &amp;</strong></td>
<td>GetShaderVariableInstance (CoreGraphics::ShaderVariableInstance semantic) const</td>
</tr>
<tr>
<td></td>
<td>get a shader variable instance</td>
</tr>
</tbody>
</table>
virtual void OnRenderBefore (IndexT frameIndex, Timing::Time time) called from ModelEntity::OnRenderBefore

void SetPosition (const Math::point & position)

const Math::point & GetPosition () const

void SetRotate (const Math::quaternion & rotate quaternion)

const Math::quaternion & GetRotate () const

void SetScale (const Math::vector & scale)

const Math::vector & GetScale () const

void SetRotatePivot (const Math::point & rotate pivot)

const Math::point & GetRotatePivot () const

void SetScalePivot (const Math::point & scale pivot)

const Math::point & GetScalePivot () const

void SetOffsetMatrix (const Math::matrix44 & m)

const Math::matrix44 & GetOffsetMatrix () const

bool IsInViewSpace () const

void SetInViewSpace (bool b)

bool GetLockedToViewer () const

void SetLockedToViewer (bool val)

const Math::matrix44 &
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetLocalTransform()</code></td>
<td>get resulting local transform matrix in local parent space</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp;</code></td>
<td><code>GetModelTransform()</code></td>
</tr>
<tr>
<td><code>virtual void DiscardHierarchy()</code></td>
<td>discard the model node instance and all of its children</td>
</tr>
<tr>
<td><code>bool IsValid()</code></td>
<td>return true if the model node instance is valid</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp;</code></td>
<td><code>GetName()</code></td>
</tr>
<tr>
<td><code>bool HasParent()</code></td>
<td>return true if node has a parent</td>
</tr>
<tr>
<td><code>const Ptr&lt;ModelNodeInstance&gt; &amp;</code></td>
<td><code>GetParent()</code></td>
</tr>
<tr>
<td><code>const Util::Array&lt; Ptr&lt;ModelNodeInstance&gt; &gt; &amp;</code></td>
<td><code>GetChildren()</code></td>
</tr>
<tr>
<td><code>bool HasChild(const Util::StringAtom &amp;name)</code></td>
<td>return true if a direct child exists by name</td>
</tr>
<tr>
<td><code>const Ptr&lt;ModelNodeInstance&gt; &amp;</code></td>
<td><code>LookupChild(const Util::StringAtom &amp;name)</code></td>
</tr>
<tr>
<td><code>Ptr&lt;ModelNodeInstance&gt;</code></td>
<td><code>LookupPath(const Util::StringAtom &amp;name)</code></td>
</tr>
<tr>
<td><code>const Ptr&lt;ModelInstance&gt; &amp;</code></td>
<td><code>GetModelInstance()</code></td>
</tr>
<tr>
<td><code>const Ptr&lt;ModelNode&gt; &amp;</code></td>
<td><code>GetModelNode()</code></td>
</tr>
<tr>
<td><code>void SetVisible(bool b, Timing::Time recursive=true)</code></td>
<td>set the node instance’s visibility</td>
</tr>
<tr>
<td><code>bool IsVisible()</code></td>
<td>return true if node instance is set to visible</td>
</tr>
<tr>
<td><code>IndexT GetModelNodeInstanceIndex()</code></td>
<td>get model node instance index for current frame</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
</tbody>
</table>
void **AddRef ()
  increment refcount by one

void **Release ()
  decrement refcount and destroy object

bool **IsInstanceOf** (const **Rtti** &rtti) const
  return true if this object is instance of given class

bool **IsInstanceOf** (const **Util::String** &className) const
  return true if this object is instance of given class by string

bool **IsInstanceOf** (const **Util::FourCC** &classFourCC) const
  return true if this object is instance of given class by fourcc

bool **IsA** (const **Rtti** &rtti) const
  return true if this object is instance of given class or derived class

bool **IsA** (const **Util::String** &rttiName) const
  return true if this object is instance of given class or derived class, by string

bool **IsA** (const **Util::FourCC** &rttiFourCC) const
  return true if this object is instance of given class or derived class, by fourcc

const **Util::String** & **GetClassName** () const
  get the class name

**Util::FourCC** **GetClassFourCC** () const
  get the class FourCC code
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>RenderDebug ()</strong></td>
</tr>
<tr>
<td></td>
<td>render node specific debug shape</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetParent (const Ptr&lt; ModelNodeInstance &gt; &amp;p)</strong></td>
</tr>
<tr>
<td></td>
<td>set parent node</td>
</tr>
<tr>
<td>void</td>
<td><strong>AddChild (const Ptr&lt; ModelNodeInstance &gt; &amp;c)</strong></td>
</tr>
<tr>
<td></td>
<td>add a child node</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnShow (Timing::Time time)</strong></td>
</tr>
<tr>
<td></td>
<td>called when the node becomes visible with current time</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnHide (Timing::Time time)</strong></td>
</tr>
<tr>
<td></td>
<td>called when the node becomes invisible</td>
</tr>
</tbody>
</table>
void Models::TransformNodeInstance::OnRenderBefore ( IndexT frameIndex, Timing::Time time ) [virtual, inherited]
called from ModelEntity::OnRenderBefore

The update method should first invoke any animators which change per-instance attributes (this is done in the parent class). Then the local space transforms must be flattened into model space.

NOTE: this method must be called late in the frame to give other systems a chance to modify the transform matrix (for instance the character attachment system).

Reimplemented from Models::ModelNodeInstance.

Reimplemented in Particles::ParticleSystemNodeInstance.

void Models::TransformNodeInstance::RenderDebug ( ) [protected, virtual, inherited]
render node specific debug shape

Render a debug visualization of the node.

Reimplemented from Models::ModelNodeInstance.

Reimplemented in Characters::CharacterNodeInstance, and Particles::ParticleSystemNodeInstance.

void Models::ModelNodeInstance::DiscardHierarchy ( ) [virtual, inherited]
discard the model node instance and all of its children

Discards this model node instance and all of its children recursively.
bool Models::ModelNodeInstance::isVisible ( ) const [inherited]

return true if node instance is set to visible

FIXME: The recursion in this is method makes it slow, especially in deep hierarchies. Also it can't be inlined.

int Core::RefCounted::getRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
| Characters::CharacterSkinSet |
Characters::CharacterSkinSet Class Reference

#include <characterskinset.h>
Detailed Description

A skin set contains the currently visible skin instances of a character instance.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CharacterSkinSet ()</code></td>
<td><code>constructor</code></td>
</tr>
<tr>
<td><code>~CharacterSkinSet ()</code></td>
<td><code>destructor</code></td>
</tr>
<tr>
<td><code>Setup (const CharacterSkinLibrary &amp;skinLib,</code></td>
<td>setup the skin set</td>
</tr>
<tr>
<td>const <code>Ptr&lt;Models::ModelInstance&gt;</code> &amp;modelInst)`</td>
<td></td>
</tr>
<tr>
<td><code>Discard ()</code></td>
<td>discard the skin set</td>
</tr>
<tr>
<td><code>IsValid () const</code></td>
<td>return true if skin set has been setup</td>
</tr>
<tr>
<td><code>AddSkin (const Util::StringAtom &amp;skinName)</code></td>
<td>add a skin to the skin set</td>
</tr>
<tr>
<td><code>RemoveSkin (const Util::StringAtom &amp;skinName)</code></td>
<td>remove a skin from the skin set</td>
</tr>
<tr>
<td><code>ToggleSkin (const Util::StringAtom &amp;skinName)</code></td>
<td>toggle skin visibility</td>
</tr>
<tr>
<td><code>Clear ()</code></td>
<td>clear all skins</td>
</tr>
<tr>
<td><code>ApplySkinList (const Util::StringAtom &amp;skinListName)</code></td>
<td>apply a skin list (clear visible skins, and make skins in the skin list visible)</td>
</tr>
<tr>
<td><code>HasSkin (const Util::StringAtom &amp;skinName)</code></td>
<td>return true if the skin set contains a skin</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>SizeT GetNumSkins () const</code></td>
<td>get number of skins in the skin set</td>
</tr>
<tr>
<td>const <code>Util::StringAtom &amp; GetSkin (IndexT index) const</code></td>
<td>get skin name by index</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Models::ModelNodeInstance &gt; &amp; GetSkinNodeInstanceByIndex (IndexT index) const</code></td>
<td>get model node instance of skin by index</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Models::ModelNodeInstance &gt; &amp; GetSkinNodeInstanceByName (const </code>Util::StringAtom &amp;name)`</td>
<td>get model node instace of skin by name</td>
</tr>
</tbody>
</table>
Characters::CharacterVariationSet
Characters::CharacterVariationSet
Class Reference

#include <charactervariationset.h>
Detailed Description

Contains the currently active variation of a character.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CharacterVariationSet()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~CharacterVariationSet()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Setup</code></td>
<td>setup the skin set (const <code>Util::StringAtom</code> &amp;defaultVariationName, const <code>CharacterVariationLibrary</code> &amp;varLib)</td>
</tr>
<tr>
<td><code>Discard</code></td>
<td>discard the skin set</td>
</tr>
<tr>
<td><code>IsValid</code></td>
<td>return true if skin set has been setup</td>
</tr>
<tr>
<td><code>SetActiveVariation</code></td>
<td>set active variation (const <code>Util::StringAtom</code> &amp;variationName)</td>
</tr>
<tr>
<td><code>GetActiveVariation</code></td>
<td>get active variation</td>
</tr>
<tr>
<td><code>GetVariationCharJointComponents</code></td>
<td>get variation transformations (const <code>Util::FixedArray&lt;CharJointComponents&gt;</code> &amp;)</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Characters::CharJointComponents
#include <charjointcomponents.h>
**Detailed Description**

Packs the components of a character joint into an array (translation, rotation, scale, variationScale) to enable packing of joint components into an array.

(C) 2009 Radon Labs GmbH
Characters::SkinnedMeshDrawInfo
Characters::SkinnedMeshDrawInfo

Class Reference

#include <skinnedmeshdrawinfo.h>

Inheritance diagram for Characters::SkinnedMeshDrawInfo:
**Detailed Description**

Front-end class for software-skinned draw info struct.

(C) 2009 Radon Labs GmbH
Characters::SkinnedMeshRenderer
#include <skinnedmeshrenderer.h>

Inheritance diagram for Characters::SkinnedMeshRenderer:
Detailed Description

Wrapper class for platform-specific skinned mesh rendering.

(C) 2008 Radon Labs GmbH
Public Types

typedef IndexT DrawHandle

*an abstract draw handle*
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SkinnedMeshRenderer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~SkinnedMeshRenderer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>GetSkinningTechnique ()</strong> const</td>
<td>get the skinning technique used by the renderer</td>
</tr>
<tr>
<td><strong>Setup ()</strong></td>
<td>setup the renderer</td>
</tr>
<tr>
<td><strong>Discard ()</strong></td>
<td>discard the renderer</td>
</tr>
<tr>
<td><strong>IsValid ()</strong> const</td>
<td>return true if renderer is valid</td>
</tr>
<tr>
<td><strong>GetRefCount ()</strong> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti)</strong> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::String &amp;className)</strong> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA (const Rtti &amp;rtti)</strong> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA (const Util::String &amp;rttiName)</strong> const</td>
<td>return true if this object is instance of given class,</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC</code></td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <strong>OnBeginFrame</strong> ()</td>
<td>call once at beginning of frame</td>
</tr>
<tr>
<td>void <strong>OnEndFrame</strong> ()</td>
<td>call once at end of frame (after rendering)</td>
</tr>
<tr>
<td>void <strong>BeginGatherSkins</strong> ()</td>
<td>begin gathering software-skinned meshes</td>
</tr>
<tr>
<td><code>void RegisterSoftwareSkinnedMesh</code> (const <code>Ptr</code>&lt;<code>Characters::CharacterInstance</code> &amp;charInst, const <code>Ptr</code>&lt;<code>CoreGraphics::Mesh</code> &amp;mesh)</td>
<td>update a software skinned mesh</td>
</tr>
<tr>
<td><code>void EndGatherSkins</code> ()</td>
<td>end gathering software-skinned meshes</td>
</tr>
<tr>
<td><code>void UpdateSoftwareSkinnedMeshes</code> ()</td>
<td>update software-skinned meshes</td>
</tr>
<tr>
<td><code>void DrawSoftwareSkinnedMesh</code> (<code>DrawHandle</code> h, <code>IndexT</code> primGroupIndex)</td>
<td>draw a software skinned mesh</td>
</tr>
<tr>
<td><code>void DrawGPUTextureSkinnedMesh</code> (const <code>Ptr</code>&lt;<code>Characters::CharacterInstance</code> &amp;charInst, const <code>Ptr</code>&lt;<code>CoreGraphics::Mesh</code> &amp;mesh, <code>IndexT</code> primGroupIndex, const <code>Ptr</code>&lt;<code>CoreGraphics::ShaderVariable</code> &amp;charInstShaderVar)</td>
<td>draw a skinned mesh</td>
</tr>
<tr>
<td><code>IndexT AllocJointTextureRow</code> ()</td>
<td>allocate a row index in the joint texture</td>
</tr>
<tr>
<td><code>void FreeJointTextureRow</code> (<code>IndexT</code> rowIndex)</td>
<td>free a row index in the joint texture</td>
</tr>
<tr>
<td><code>void * AcquireJointTextureRowPointer</code> (const <code>Ptr</code>&lt;<code>Characters::CharacterInstance</code> &amp;charInst, <code>SizeT</code> &amp;outRowPitch)</td>
<td>get a pointer to the joint texture row for the given character instance</td>
</tr>
</tbody>
</table>
Member Function Documentation

**SkinnedMeshRendererBase::DrawHandle**

Base::SkinnedMeshRendererBase::RegisterSoftwareSkinnedMesh(const Ptr<Characters::CharacterInstance> & &
const Ptr<CoreGraphics::Mesh> & &
)

update a software skinned mesh

This method should only be called when RequiresSoftwareSkinning() returns true!

This registers a mesh for software-skinning in the
**UpdateSoftwareSkinnedMeshes()** which must be called after
**EndGatherSkins()**.

This method may be called more then once per character-instance/mesh combination! The method will drop duplicates.

**void**

Base::SkinnedMeshRendererBase::UpdateSoftwareSkinnedMeshes() [protected, inherited]

update software-skinned meshes

On platforms with software-skinning, this method should perform the skinning for all meshes gathered during the GatherSkins pass.

**void**

Base::SkinnedMeshRendererBase::DrawSoftwareSkinnedMesh(DrawHandle h,
IndexT primGroupIndex)

[protected, inherited]

draw a software skinned mesh

This method should only be called when RequiresSoftwareSkinning() returns true!
Software-skinning platforms call this method with the DrawHandle returned by UpdateSoftwareSkinnedMesh() to draw a portion of the skinned mesh:

```cpp
template<typename T>
void Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
template<typename T>
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
template<typename T>
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
template<typename T>
const Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
template<typename T>
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
template<typename T>
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```


dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Characters::SkinningTechnique
Characters::SkinningTechnique Class Reference

#include <skinningtechnique.h>
Detailed Description

The various character skinning techniques supported by N3.

(C) 2009 Radon Labs GmbH
Commands::CreateEntityCommand
Commands::CreateEntityCommand
Class Reference

#include <createentitycommand.h>
Detailed Description

(C) 2010 Radon Labs GmbH
Public Member Functions

- **CreateEntityCommand()**  
  *constructor*

- **virtual ~CreateEntityCommand()**  
  *destructor*
## Protected Member Functions

<table>
<thead>
<tr>
<th>virtual void</th>
<th><strong>HandleCommand</strong> (const Util::CommandLineArgs &amp;cmd)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>handle command</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>HandleDefaultEntity</strong> (const Util::String &amp;category, const Util::String &amp;tmpl, const Util::String &amp;id, const Math::matrix44 &amp;entityTransform, const Attr::AttributeContainer &amp;attrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>handle update of a default game entity</em></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:41 2010
Conditions:

- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Conditions: And
#include <and.h>

Inheritance diagram for Conditions::And:
Detailed Description

Returns the logical AND result of a list of conditions.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool Evaluate  ()</td>
<td>Check whether the condition is true</td>
</tr>
<tr>
<td>virtual void Assert ()</td>
<td>Assert that all required data is present in the world database</td>
</tr>
<tr>
<td>virtual bool Assert (const Ptr<a href="">Script::InfoLog</a> &amp;infoLog)</td>
<td>Like <code>Assert()</code> but adds errors to the info log object instead of closing the application</td>
</tr>
<tr>
<td>void AddCondition (const Ptr<a href="">Conditions::Condition</a> &amp;condition)</td>
<td>Add one condition</td>
</tr>
<tr>
<td>const Util::Array&lt; Ptr<a href="">Conditions::Condition</a> &gt; &amp; GetConditions () const</td>
<td>Get a list of conditions</td>
</tr>
<tr>
<td>virtual void SetEntity (const Ptr<a href="">Game::Entity</a> &amp;entity)</td>
<td>Set optional entity</td>
</tr>
<tr>
<td>virtual void Write (const Ptr<a href="">Script::ActionReader</a> &amp;actionReader)</td>
<td>Write to action reader</td>
</tr>
<tr>
<td>virtual void Read (const Ptr<a href="">Script::ActionReader</a> &amp;actionReader)</td>
<td>Read from action reader</td>
</tr>
<tr>
<td>virtual void ParseArgs (const Util::CommandLineArgs &amp;args)</td>
<td>Parse arguments from command line args object</td>
</tr>
<tr>
<td>const Ptr<a href="">Game::Entity</a> &amp; GetEntity () const</td>
<td>Get entity</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>HasEntity () const</code></td>
<td>Get optional target entity if exists.</td>
</tr>
<tr>
<td><code>bool HasEntity () const</code></td>
<td>Does this contain a target entity?</td>
</tr>
<tr>
<td><code>void SetResult (bool b)</code></td>
<td>set the condition's result, set by msg handler which may handle this</td>
</tr>
<tr>
<td><code>bool GetResult () const</code></td>
<td>get the set result, may be called by subclasses only</td>
</tr>
<tr>
<td><code>bool CheckId (const Messaging::Id &amp;id) const</code></td>
<td>return true if message is of the given id</td>
</tr>
<tr>
<td><code>virtual void Encode (const Ptr&lt;IO::BinaryWriter&gt; &amp;writer)</code></td>
<td>encode message into a stream</td>
</tr>
<tr>
<td><code>virtual void Decode (const Ptr&lt;IO::BinaryReader&gt; &amp;reader)</code></td>
<td>decode message from a stream</td>
</tr>
<tr>
<td><code>void SetHandled (bool b)</code></td>
<td>set the handled flag</td>
</tr>
<tr>
<td><code>bool Handled () const</code></td>
<td>return true if the message has been handled</td>
</tr>
<tr>
<td><code>void SetDeferred (bool b)</code></td>
<td>set deferred flag</td>
</tr>
<tr>
<td><code>bool IsDeferred () const</code></td>
<td>get deferred flag</td>
</tr>
<tr>
<td><code>void SetDeferredHandled (bool b)</code></td>
<td>set the deferred handled flag</td>
</tr>
<tr>
<td><code>bool DeferredHandled () const</code></td>
<td>get the deferred handled flag</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>increment refcount by one</td>
<td></td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (constUtil::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName() const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC() const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Ptr&lt; Condition &gt;</code></td>
<td><code>CreateConditionFromString</code> (const <code>Util::String</code> &amp;cmd)</td>
<td>create a complete condition object from a string</td>
</tr>
<tr>
<td>static <code>Util::Array&lt; Ptr&lt; Condition &gt; &gt;</code></td>
<td><code>CreateConditionsFromString</code> (const <code>Util::String</code> &amp;cmd)</td>
<td>create several conditions from semicolon-separated string</td>
</tr>
<tr>
<td>static void</td>
<td><code>DumpRefCountingLeaks</code> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Ptr< Condition >**

Conditions::Condition::CreateConditionFromString (Const Util::String cmd ) [static, inherited]

create a complete condition object from a string

Static method which creates any condition object from a command string of the form:

cmd key0=value0 key1=value1 key2=value2

**Util::Array< Ptr< Condition > >**

Conditions::Condition::CreateConditionsFromString (Const Util::String cmd ) [static, inherited]

create several conditions from semicolon-separated string

Static method which creates many action from a string of the form accepted by CreateConditionFromString() where several conditions are separated by a semicolon.

void

Conditions::Condition::ParseArgs (Const Util::CommandLineArgs args ) [virtual, inherited]

parse arguments from command line args object

This method should initialize the Condition object from a CommandLineArgs object. Override in subclass.

int

Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void

Core::RefCounted::AddRef ( ) [inline, inherited]
Increment the refcount of the object.

```c
void
Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```c
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```c
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```c
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Conditions:: :Condition

Conditions::Condition Class Reference

#include <condition.h>

Inheritance diagram for Conditions::Condition:

```
  Core::RefCounted
   "   "  "   "  "   "  "   "  "   "
  "   "  "   "  "   "  "   "  "   "
  Messaging::Message
   "   "  "   "  "   "  "   "  "   "
  "   "  "   "  "   "  "   "  "   "
  Conditions::Condition
   "   "  "   "  "   "  "   "  "   "
  "   "  "   "  "   "  "   "  "   "
  Conditions::And
   "   "  "   "  "   "  "   "  "   "
  "   "  "   "  "   "  "   "  "   "
  Conditions::FSMCondition
   "   "  "   "  "   "  "   "  "   "
  "   "  "   "  "   "  "   "  "   "
  Conditions::Not
   "   "  "   "  "   "  "   "  "   "
  "   "  "   "  "   "  "   "  "   "
  Conditions::Or
```


Detailed Description

The base class for conditions. Conditions and Actions are the basic building blocks of world interaction. Conditions are small C++ objects which check whether some condition in the world is true. They are used by several other subsystems, like the quest and dialog system.

(C) 2005 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Condition</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual void Assert ()</td>
<td>Assert correct initialisation</td>
</tr>
<tr>
<td>virtual bool Assert (const Ptr&lt; Script::InfoLog &gt; &amp;infoLog)</td>
<td>like Assert() but adds errors to the info log object instead of closing the application</td>
</tr>
<tr>
<td>virtual void ParseArgs (const Util::CommandLineArgs &amp;args)</td>
<td>parse arguments from command line args object</td>
</tr>
<tr>
<td>virtual bool Evaluate ()</td>
<td>Internal evaluation</td>
</tr>
<tr>
<td>virtual void SetEntity (const Ptr&lt; Game::Entity &gt; &amp;v)</td>
<td>Set optional entity to <code>v</code>.</td>
</tr>
<tr>
<td>const Ptr&lt; Game::Entity &gt; &amp; GetEntity () const</td>
<td>Get optional target entity if exists.</td>
</tr>
<tr>
<td>bool HasEntity () const</td>
<td>Does this contain a target entity?</td>
</tr>
<tr>
<td>void SetResult (bool b)</td>
<td>Set the condition's result, set by msg handler which may handle this</td>
</tr>
<tr>
<td>bool GetResult () const</td>
<td>Get the set result, may be called by subclasses only</td>
</tr>
<tr>
<td>virtual void Write (const Ptr&lt; Script::ActionReader &gt; &amp;actionReader)</td>
<td>Write to action reader</td>
</tr>
<tr>
<td>virtual void Read (const Ptr&lt; Script::ActionReader &gt; &amp;actionReader)</td>
<td>Read from action reader</td>
</tr>
<tr>
<td>bool CheckId (const Messaging::Id &amp;id) const</td>
<td>Return true if message is of the given id</td>
</tr>
<tr>
<td>virtual void Encode (const Ptr&lt; IO::BinaryWriter &gt; &amp;writer)</td>
<td>Encode the condition into the given binary writer.</td>
</tr>
</tbody>
</table>
virtual void Decode (const Ptr< IO::BinaryReader >& reader)

decode message from a stream

void SetHandled (bool b)

set the handled flag

bool Handle () const

return true if the message has been handled

void SetDeferred (bool b)

set deferred flag

bool IsDeferred () const

get deferred flag

void SetDeferredHandled (bool b)

set the deferred handled flag

bool DeferredHandled () const

get the deferred handled flag

int GetRefCount () const

get the current refcount

void AddRef ()

increment refcount by one

void Release ()

decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const

return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const

return true if this object is instance of given class, by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const

return true if this object is instance of given class, by fourcc

bool IsA (const Rtti &rtti) const

return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const

return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const

return true if this object is instance of given class, or a derived class, by fourcc.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Util::String &amp; GetClassName()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC()</code></td>
<td>const</td>
</tr>
</tbody>
</table>

return true if this object is instance of given class, or a derived class, by fourcc
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Static Public Member Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static</strong> <code>Ptr&lt; Condition &gt;</code> <code>CreateConditionFromString</code> (const <code>Util::String</code> &amp;cmd)</td>
</tr>
<tr>
<td>create a complete condition object from a string</td>
</tr>
<tr>
<td><strong>static</strong> <code>Util::Array&lt; Ptr&lt; Condition &gt; &gt;</code> <code>CreateConditionsFromString</code> (const <code>Util::String</code> &amp;cmd)</td>
</tr>
<tr>
<td>create several conditions from semicolon-separated string</td>
</tr>
<tr>
<td><strong>static void</strong> <code>DumpRefCountingLeaks</code> ()</td>
</tr>
<tr>
<td>dump refcounting leaks, call at end of application</td>
</tr>
<tr>
<td>(NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Ptr< Condition >**

Conditions::Condition::CreateConditionFromString ( Util::String cmd ) [static]

create a complete condition object from a string

Static method which creates any condition object from a command string of the form:

cmd key0=value0 key1=value1 key2=value2

**Util::Array< Ptr< Condition > >**

Conditions::Condition::CreateConditionsFromString ( Util::String cmd ) [static]

create several conditions from semicolon-separated string

Static method which creates many action from a string of the form accepted by CreateConditionFromString() where several conditions are separated by a semicolon.

```
void
Conditions::Condition::Assert ( ) [virtual]
```

assert correct initialisation

This method asserts that all data required by the conditions actually exists. Override in subclass!

Reimplemented in Conditions::And, Conditions::Not, and Conditions::Or.

```
bool
Conditions::Condition::Assert ( const Ptr< ( Script::InfoLog infoLog ) [virtual] > &
```

like Assert() but adds errors to the info log object instead of closing the application
This method asserts that all data actually required by the conditions exist. In subclasses errors can be added to the info log object and in case of errors false can be returned instead of closing the application.

Override in subclass! (infoLog in this class is ignored; Assert() will be called; returns always true)

Reimplemented in Conditions::And, Conditions::Not, and Conditions::Or.

```cpp
void Conditions::Condition::ParseArgs ( const Util::CommandLineArgs & args ) [virtual]
```

parse arguments from command line args object

This method should initialize the Condition object from a CommandLineArgs object. Override in subclass.

```cpp
bool Conditions::Condition::Evaluate () [virtual]
```

internal evaluation

This method evaluates whether the condition is true, Negates result if not flag is set.

Reimplementeded in Conditions::And, Conditions::Not, and Conditions::Or.

```cpp
int Core::RefCounted::GetRefCount () const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef () [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
### Conditions

- **Main Page**
- **Namespaces**
- **Data Structures**
- **Files**
- **Related Pages**

- **Alphabetical List**
- **Data Structures**
- **Class Hierarchy**
- **Data Fields**

**Conditions::** `FSMCondition`
#include <fsmcondition.h>

Inheritance diagram for Conditions::FSMCondition:
Detailed Description

A specialized condition for the finite state machine system which is activated when its state becomes active and which gets notified about messages to the parent entity. This enabled the condition to be triggered by incoming messages instead of a polling mechanism.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <code>Notify</code> (const <code>Ptr&lt; Messaging::Message &gt;</code> &amp;msg)</td>
<td>notify about incoming message</td>
</tr>
<tr>
<td>virtual void <code>OnActivate</code> ()</td>
<td>called on state activation</td>
</tr>
<tr>
<td>virtual void <code>OnDeactivate</code> ()</td>
<td>called on state deactivation</td>
</tr>
<tr>
<td>virtual void <code>Assert</code> ()</td>
<td>assert correct initialisation</td>
</tr>
<tr>
<td>virtual bool <code>Assert</code> (const <code>Ptr&lt; Script::InfoLog &gt;</code> &amp;infoLog)</td>
<td>like <code>Assert()</code> but adds errors to the info log object instead of closing the application</td>
</tr>
<tr>
<td>virtual void <code>ParseArgs</code> (const <code>Util::CommandLineArgs</code> &amp;args)</td>
<td>parse arguments from command line args object</td>
</tr>
<tr>
<td>virtual bool <code>Evaluate</code> ()</td>
<td>internal evaluation</td>
</tr>
<tr>
<td>virtual void <code>SetEntity</code> (const <code>Ptr&lt; Game::Entity &gt;</code> &amp;v)</td>
<td>Set optional entity to <code>v</code>.</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Game::Entity &gt;</code> &amp; <code>GetEntity</code> ()</td>
<td>Get optional target entity if exists.</td>
</tr>
<tr>
<td>bool <code>HasEntity</code> ()</td>
<td>Does this contain a target entity?</td>
</tr>
<tr>
<td>void <code>SetResult</code> (bool b)</td>
<td>set the condition's result, set by msg handler which may handle this</td>
</tr>
<tr>
<td>bool <code>GetResult</code> ()</td>
<td>get the set result, may be called by subclasses only</td>
</tr>
<tr>
<td>virtual void <code>Write</code> (const <code>Ptr&lt; Script::ActionReader &gt;</code> &amp;actionReader)</td>
<td>write to action reader</td>
</tr>
<tr>
<td>virtual void <code>Read</code> (const <code>Ptr&lt; Script::ActionReader &gt;</code> &amp;actionReader)</td>
<td></td>
</tr>
</tbody>
</table>
bool CheckId (const Messaging::Id &id) const
return true if message is of the given id

virtual void Encode (const Ptr< IO::BinaryWriter > &writer)
encode message into a stream

virtual void Decode (const Ptr< IO::BinaryReader > &reader)
decode message from a stream

void SetHandled (bool b)
set the handled flag

bool Handled () const
return true if the message has been handled

void SetDeferred (bool b)
set deferred flag

bool IsDeferred () const
get deferred flag

void SetDeferredHandled (bool b)
set the deferred handled flag

bool DeferredHandled () const
get the deferred handled flag

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static <code>Ptr&lt; Condition &gt;</code></th>
<th><code>CreateConditionFromString</code> (const <code>Util::String</code> &amp;cmd)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>create a complete condition object from a string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static <code>Util::Array&lt; Ptr&lt; Condition &gt; &gt;</code></th>
<th><code>CreateConditionsFromString</code> (const <code>Util::String</code> &amp;cmd)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>create several conditions from semicolon-separated string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static void</th>
<th><code>DumpRefCountingLeaks</code> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Ptr< Condition >**

Conditions::Condition::CreateConditionFromString (Util::String cmd) [static, inherited]

create a complete condition object from a string

Static method which creates any condition object from a command string of the form:

cmd key0=value0 key1=value1 key2=value2

**Util::Array< Ptr< Condition > >**

Conditions::Condition::CreateConditionsFromString (Util::String cmd) [static, inherited]

create several conditions from semicolon-separated string

Static method which creates many action from a string of the form accepted by CreateConditionFromString() where several conditions are separated by a semicolon.

**void**

Conditions::Condition::Assert () [virtual, inherited]

assert correct initialisation

This method asserts that all data required by the conditions actually exists. Override in subclass!

Reimplemented in Conditions::And, Conditions::Not, and Conditions::Or.

**bool**

Conditions::Condition::Assert (const Ptr< (Script::InfoLog infoLog) > & infoLog) [virtual, inherited]

like Assert() but adds errors to the info log object instead of closing the application
This method asserts that all data actually required by the conditions exist. In subclasses errors can be added to the info log object and in case of errors false can be returned instead of closing the application.

Override in subclass! (infoLog in this class is ignored; Assert() will be called; returns always true)

Reimplemented in Conditions::And, Conditions::Not, and Conditions::Or.

```cpp
void Conditions::Condition::ParseArgs(const Util::CommandLineArgs& args) [virtual, inherited]
```

parse arguments from command line args object

This method should initialize the Condition object from a CommandLineArgs object. Override in subclass.

```cpp
bool Conditions::Condition::Evaluate() [virtual, inherited]
```

internal evaluation

This method evaluates whether the condition is true, Negates result if not flag is set.

Reimplemented in Conditions::And, Conditions::Not, and Conditions::Or.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Conditions:: Not
Conditions::Not Class Reference

#include <not.h>

Inheritance diagram for Conditions::Not:

```
  Core::RefCounted
     ↓
  Messaging::Message
     ↓
Conditions::Condition
     ↓
Conditions::Not
```
Detailed Description

Returns the inverted result of a given condition.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool <code>Evaluate()</code></td>
<td>check whether the condition is true</td>
</tr>
<tr>
<td>virtual void <code>Assert()</code></td>
<td>assert that all required data is present in the world database</td>
</tr>
<tr>
<td>virtual bool <code>Assert(const Ptr&lt; Script::InfoLog &gt;&amp; infoLog)</code></td>
<td>like <code>Assert()</code> but adds errors to the info log object instead of closing the application</td>
</tr>
<tr>
<td>void <code>SetCondition(const Ptr&lt; Conditions::Condition &gt;&amp; condition)</code></td>
<td>set the condition</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Conditions::Condition &gt;&amp; GetCondition()</code> const</td>
<td>get the condition</td>
</tr>
<tr>
<td>virtual void <code>SetEntity(const Ptr&lt; Game::Entity &gt;&amp; entity)</code></td>
<td>set optional entity to entity</td>
</tr>
<tr>
<td>virtual void <code>Write(const Ptr&lt; Script::ActionReader &gt;&amp; actionReader)</code></td>
<td>write to action reader</td>
</tr>
<tr>
<td>virtual void <code>Read(const Ptr&lt; Script::ActionReader &gt;&amp; actionReader)</code></td>
<td>read from action reader</td>
</tr>
<tr>
<td>virtual void <code>ParseArgs(const Util::CommandLineArgs &amp;args)</code></td>
<td>parse arguments from command line args object</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Game::Entity &gt;&amp; GetEntity()</code> const</td>
<td>Get optional target entity if exists.</td>
</tr>
<tr>
<td>bool <code>HasEntity()</code> const</td>
<td>Does this contain a target entity?</td>
</tr>
<tr>
<td>void <code>SetResult(bool b)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool GetResult()</td>
<td>get the set result, may be called by subclasses only</td>
</tr>
<tr>
<td>bool CheckId(const Messaging::Id &amp;id)</td>
<td>return true if message is of the given id</td>
</tr>
<tr>
<td>virtual void Encode(const Ptr<a href="">IO::BinaryWriter</a> &amp;writer)</td>
<td>encode message into a stream</td>
</tr>
<tr>
<td>virtual void Decode(const Ptr<a href="">IO::BinaryReader</a> &amp;reader)</td>
<td>decode message from a stream</td>
</tr>
<tr>
<td>void SetHandled(bool b)</td>
<td>set the handled flag</td>
</tr>
<tr>
<td>bool Handled() const</td>
<td>return true if the message has been handled</td>
</tr>
<tr>
<td>void SetDeferred(bool b)</td>
<td>set deferred flag</td>
</tr>
<tr>
<td>bool IsDeferred() const</td>
<td>get deferred flag</td>
</tr>
<tr>
<td>void SetDeferredHandled(bool b)</td>
<td>set the deferred handled flag</td>
</tr>
<tr>
<td>bool DeferredHandled() const</td>
<td>get the deferred handled flag</td>
</tr>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>Util::String</strong> &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
# Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static Ptr&lt; Condition &gt; CreateConditionFromString (const Util::String &amp;cmd)</code></td>
<td>create a complete condition object from a string</td>
</tr>
<tr>
<td><code>static Util::Array&lt; Ptr&lt; Condition &gt; &gt; CreateConditionsFromString (const Util::String &amp;cmd)</code></td>
<td>create several conditions from semicolon-separated string</td>
</tr>
<tr>
<td><code>static void DumpRefCountingLeaks ()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Ptr< Condition >**

`Conditions::Condition::CreateConditionFromString (Util::String cmd ) [static, inherited]`

create a complete condition object from a string

Static method which creates any condition object from a command string of the form:

`cmd key0=value0 key1=value1 key2=value2`

**Util::Array< Ptr< Condition > >**

`Conditions::Condition::CreateConditionsFromString (Util::String cmd ) [static, inherited]`

create several conditions from semicolon-separated string

Static method which creates many action from a string of the form accepted by `CreateConditionFromString()` where several conditions are separated by a semicolon.

**void**

`Conditions::Condition::ParseArgs (Util::CommandLineArgs args ) [virtual, inherited]`

parse arguments from command line args object

This method should initialize the `Condition` object from a CommandLineArgs object. Override in subclass.

**int**

`Core::RefCounted::GetRefCount ( ) const [inline, inherited]`

get the current refcount

Return the current refcount of the object.

**void**

`Core::RefCounted::AddRef ( ) [inline, inherited]`
increment refcount by one

Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Conditions::Operator
Conditions::Operator< TYPE > Class Template Reference

#include <operator.h>
Detailed Description

template<class TYPE>
class Conditions::Operator< TYPE >

Template operator class for numeric compare operations in scripts.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operator ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>Operator (OpType type)</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>bool <strong>Evaluate</strong> (TYPE first, TYPE second) const</td>
<td>operator evaluation</td>
</tr>
<tr>
<td>void <strong>FromString</strong> (const Util::String &amp;str)</td>
<td>set type from string</td>
</tr>
</tbody>
</table>
#include <or.h>

Inheritance diagram for Conditions::Or:
Detailed Description

Returns the logical OR result of a list of conditions.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual bool Evaluate()</code></td>
<td>check whether the condition is true</td>
</tr>
<tr>
<td><code>virtual void Assert()</code></td>
<td>assert that all required data is present in the world database</td>
</tr>
<tr>
<td><code>virtual bool Assert(const Ptr&lt;Script::InfoLog&gt; &amp;infoLog)</code></td>
<td>like <code>Assert()</code> but adds errors to the info log object instead of closing the application</td>
</tr>
<tr>
<td><code>void AddCondition(const Ptr&lt;Conditions::Condition&gt; &amp;condition)</code></td>
<td>add one condition</td>
</tr>
<tr>
<td><code>const Util::Array&lt;Ptr&lt;Conditions::Condition&gt;&gt; &amp; GetConditions()</code></td>
<td>get a list of conditions</td>
</tr>
<tr>
<td><code>virtual void Write(const Ptr&lt;Script::ActionReader&gt; &amp;actionReader)</code></td>
<td>write to action reader</td>
</tr>
<tr>
<td><code>virtual void Read(const Ptr&lt;Script::ActionReader&gt; &amp;actionReader)</code></td>
<td>read from action reader</td>
</tr>
<tr>
<td><code>virtual void SetEntity(const Ptr&lt;Game::Entity&gt; &amp;entity)</code></td>
<td>set optional entity</td>
</tr>
<tr>
<td><code>virtual void ParseArgs(const Util::CommandLineArgs &amp;args)</code></td>
<td>parse arguments from command line args object</td>
</tr>
<tr>
<td><code>const Ptr&lt;Game::Entity&gt; &amp; GetEntity()</code></td>
<td>get entity</td>
</tr>
</tbody>
</table>
Get optional target entity if exists.

```cpp
bool HasEntity() const

Does this contain a target entity?
```

```cpp
void SetResult(bool b)

set the condition's result, set by msg handler which may handle this
```

```cpp
bool GetResult() const

get the set result, may be called by subclasses only
```

```cpp
bool CheckId(const Messaging::Id &id) const

return true if message is of the given id
```

```cpp
virtual void Encode(const Ptr<IO::BinaryWriter> &writer)

encode message into a stream
```

```cpp
virtual void Decode(const Ptr<IO::BinaryReader> &reader)

decode message from a stream
```

```cpp
void SetHandled(bool b)

set the handled flag
```

```cpp
bool Handled() const

return true if the message has been handled
```

```cpp
void SetDeferred(bool b)

set deferred flag
```

```cpp
bool IsDeferred() const

get deferred flag
```

```cpp
void SetDeferredHandled(bool b)

set the deferred handled flag
```

```cpp
bool DeferredHandled() const

get the deferred handled flag
```

```cpp
int GetRefCount() const

get the current refcount
```

```cpp
void AddRef()
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>increment refcount by one</code></td>
<td>void <code>Release ()</code></td>
</tr>
<tr>
<td></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC <code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static </code></td>
<td><code>Ptr&lt; Condition &gt;</code></td>
<td><strong>CreateConditionFromString</strong> (const <strong>Util::String</strong> &amp;cmd)</td>
</tr>
<tr>
<td>static <code>Util::Array&lt; Ptr&lt; Condition &gt; &gt;</code></td>
<td><code>CreateConditionsFromString</code> (const <strong>Util::String</strong> &amp;cmd)</td>
<td></td>
</tr>
<tr>
<td>static void</td>
<td><code>DumpRefCountingLeaks</code> ()</td>
<td></td>
</tr>
</tbody>
</table>

create a complete condition object from a string
create several conditions from semicolon-separated string
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

**Ptr< Condition >**

Conditions::Condition::CreateConditionFromString (Const Util::String cmd ) [static, inherited]

Create a complete condition object from a string

Static method which creates any condition object from a command string of the form:

cmd key0=value0 key1=value1 key2=value2

**Util::Array< Ptr< Condition >>**

Conditions::Condition::CreateConditionsFromString (Const Util::String cmd ) [static, inherited]

Create several conditions from semicolon-separated string

Static method which creates many action from a string of the form accepted by CreateConditionFromString() where several conditions are separated by a semicolon.

**void**

Conditions::Condition::ParseArgs (Const Util::CommandLineArgs args ) [virtual, inherited]

Parse arguments from command line args object

This method should initialize the Condition object from a CommandLineArgs object. Override in subclass.

**int**

Core::RefCounted::GetRefCount () const [inline, inherited]

Get the current refcount

Return the current refcount of the object.

**void**

Core::RefCounted::AddRef () [inline, inherited]
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields
core Class Reference

#include </types.h>
Detailed Description

Basic type definitions for Nebula3.

(C) 2006 Radon Labs GmbH
Core::CoreServer
Core::CoreServer Class Reference

#include <coreserver.h>

Inheritance diagram for Core::CoreServer:
Detailed Description

The central core server object initializes a minimal Nebula3 runtime environment necessary to boot up the rest of Nebula3. It should be the first object a Nebula3 application creates, and the last to destroy before shutdown.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CoreServer()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~CoreServer()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>void SetCompanyName(const Util::StringAtom &amp;s)</code></td>
<td>set the company name</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetCompanyName()</code></td>
<td>get the company name</td>
</tr>
<tr>
<td><code>void SetAppName(const Util::StringAtom &amp;s)</code></td>
<td>set the application name</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetAppName()</code></td>
<td>get the application name</td>
</tr>
<tr>
<td><code>void SetRootDirectory(const Util::StringAtom &amp;s)</code></td>
<td>set the root directory of the application (default is “home:”)</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetRootDirectory()</code></td>
<td>get the root directory of the application</td>
</tr>
<tr>
<td><code>void Open()</code></td>
<td>open the core server</td>
</tr>
<tr>
<td><code>void Close()</code></td>
<td>close the core server</td>
</tr>
<tr>
<td><code>bool IsOpen()</code></td>
<td>return true if currently open</td>
</tr>
<tr>
<td><code>void Trigger()</code></td>
<td>trigger core server, updates console</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

gle the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Core::ExitHandler
Core::ExitHandler Class Reference

#include <exithandler.h>
**Detailed Description**

ExitHandlers are static objects which register themselves automatically once at startup and are called back from the Core::SysFunc::Exit() static method which is called right before a Nebula3 application exists. Please note that the Nebula3 runtime usually doesn't yet exist when the ExitHandler is created or destroyed, so don't put anything complex into the constructor or destructor of the class!

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ExitHandler ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~ExitHandler ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnExit ()</code> const</td>
</tr>
<tr>
<td>const <code>ExitHandler * Next ()</code></td>
<td>get pointer to next exit handler in forward linked list</td>
</tr>
<tr>
<td></td>
<td>virtual method called from SysFunc::Exit()</td>
</tr>
</tbody>
</table>
Core::Factory
Core::Factory Class Reference

#include <factory.h>
Detailed Description

Provides the central object factory mechanism for Nebula3. Classes which are derived from RefCounted register themselves automatically to the central Factory object through the __DeclareClass and __ImplementClass macros.

(C) 2005 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>Register</strong> (const <strong>Rtti</strong> *rtti, const <strong>Util::String</strong> &amp;className, const <strong>Util::FourCC</strong> &amp;classFourCC)</td>
<td>register a RTTI object with the factory (with class name and class fourcc code)</td>
</tr>
<tr>
<td>void</td>
<td><strong>Register</strong> (const <strong>Rtti</strong> *rtti, const <strong>Util::String</strong> &amp;className)</td>
<td>register a RTTI object with the factory (without fourcc code)</td>
</tr>
<tr>
<td>bool</td>
<td><strong>ClassExists</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td>check if a class exists by class name</td>
</tr>
<tr>
<td>bool</td>
<td><strong>ClassExists</strong> (const <strong>Util::FourCC</strong> classFourCC) const</td>
<td>check if a class exists by FourCC code</td>
</tr>
<tr>
<td>const <strong>Rtti</strong> *</td>
<td><strong>GetClassRtti</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td>get class rtti object by name</td>
</tr>
<tr>
<td>const <strong>Rtti</strong> *</td>
<td><strong>GetClassRtti</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
<td>get class rtti object by fourcc code</td>
</tr>
<tr>
<td><strong>RefCounted</strong> *</td>
<td><strong>Create</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td>create an object by class name</td>
</tr>
<tr>
<td><strong>RefCounted</strong> *</td>
<td><strong>Create</strong> (const <strong>Util::FourCC</strong> classFourCC) const</td>
<td>create an object by FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>Factory * Instance ()</td>
<td>get pointer to singleton instance (cannot use singleton.h!)</td>
</tr>
<tr>
<td>static void</td>
<td>Destroy ()</td>
<td>static instance destruction method</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Factory**

```cpp
Core::Factory::Instance() [static]
```

get pointer to singleton instance (cannot use singleton.h!)

The factory's constructor is called by the **Instance()** method, and nobody else.

```cpp
void Core::Factory::Destroy() [static]
```

static instance destruction method

This static method is used to destroy the factory object and should be called right before the main function exits. It will make sure that no accidental memory leaks are reported by the debug heap.

```cpp
void Core::Factory::Register(const Rtti * rti,
const Util::String className,
const Util::FourCC classFourCC)
```

register a RTTI object with the factory (with class name and class fourcc code)

This registers an **Rtti** object with the factory and is called from an **Rtti** object's constructor. The function will fail with an error message box if a class with an identical class name or fourcc code has already been registered.

NOTE: we cannot use the class name of fourcc from the RTTI object, because it may be that the RTTI object hasn't been initialized yet when this method is called (initialization order of global variables is
void Core::Factory::Register ( const Rtti * rtti, const Util::String className & )

register a RTTI object with the factory (without fourcc code)

Legacy version of Register() which doesn't require a class fourcc code. This is for compatibility with old Mangalore applications.

bool Core::Factory::ClassExists ( const Util::String className & ) const

check if a class exists by class name

This method checks if a class with the given name has been registered.

bool Core::Factory::ClassExists ( const Util::FourCC classFourCC ) const

check if a class exists by FourCC code

This method checks if a class with the given fourcc code has been registered.

RefCounted * Core::Factory::Create ( const Util::String className & ) const

create an object by class name

Create an object by class name.

RefCounted * Core::Factory::Create ( const Util::FourCC classFourCC ) const

create an object by FourCC code
Create an object by FourCC code.
Core::RefCounted
Core::RefCounted Class Reference

#include <refcounted.h>

Inheritance diagram for Core::RefCounted:
Detailed Description

The common base class of Nebula3. Implement a strong refcounted mechanism and runtime type information. Nebula3 checks at application shutdown for proper cleanup of all `RefCounted` objects. Refcounting leaks will generate a log on the debug output.

FIXME: The `RefCounted` class uses Interlocked functions and a CriticalSection to guarantee thread-safe refcounting and destruction, but only some classes need this (mostly messages which are passed between threads). If it is guaranteed that an object is only manipulated from the same thread then this thread-synchronization adds unnecessary overhead.

(C) 2006 RadonLabs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RefCounted()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti)</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName)</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName()</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC()</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

virtual ~RefCounted ()

destructor (called when refcount reaches zero)
**Member Function Documentation**

```cpp
int Core::RefCounted::GetRefCount() const [inline]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Core::RefCountedList
Core::RefCountedList Class Reference

#include <refcountedlist.h>

Inheritance diagram for Core::RefCountedList:

Util::List< TYPE >

< TYPE >" shape="rect" coords="0,0,131,24">
Detailed Description

Implements a static list which keeps track of all refcounted objects to detect refcounting leaks at application shutdown. Will only be active when the application is compiled in debug mode.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void DumpLeaks()</code></td>
<td>dump memory leaks, this method is called by <code>RefCounted::DumpRefCountedLeaks()</code></td>
</tr>
<tr>
<td><code>bool IsEmpty()</code></td>
<td>return <code>true</code> if the list is empty</td>
</tr>
<tr>
<td><code>SizeT Size()</code></td>
<td>get number of elements in list (slow)</td>
</tr>
<tr>
<td><code>void Clear()</code></td>
<td>clear list</td>
</tr>
<tr>
<td><code>void AddList(const List&lt;TYPE&gt; &amp;l)</code></td>
<td>add contents of other list to this list</td>
</tr>
<tr>
<td><code>Iterator AddAfter(Iterator iter, const TYPE &amp;e)</code></td>
<td>add element after given element</td>
</tr>
<tr>
<td><code>Iterator AddBefore(Iterator iter, const TYPE &amp;e)</code></td>
<td>add element before given element</td>
</tr>
<tr>
<td><code>Iterator AddFront(const TYPE &amp;e)</code></td>
<td>add element to beginning of list</td>
</tr>
<tr>
<td><code>Iterator AddBack(const TYPE &amp;e)</code></td>
<td>add element to end of list</td>
</tr>
<tr>
<td><code>TYPE RemoveFront()</code></td>
<td>remove first element of list</td>
</tr>
<tr>
<td><code>TYPE RemoveBack()</code></td>
<td>remove last element of list</td>
</tr>
<tr>
<td><code>TYPE Remove(Iterator iter)</code></td>
<td>remove given element</td>
</tr>
<tr>
<td><code>TYPE &amp; Front()</code></td>
<td>get first element</td>
</tr>
<tr>
<td><code>TYPE &amp; Back()</code></td>
<td>get last element</td>
</tr>
<tr>
<td><code>Iterator Begin()</code></td>
<td>get iterator to first element</td>
</tr>
<tr>
<td><code>Iterator End()</code></td>
<td>get iterator past the last element</td>
</tr>
</tbody>
</table>
Iterator **Find** (const TYPE &e, Iterator start) const

*find element in array (slow)*
Core::Rtti
#include <rtti.h>
Detailed Description

Nebula3's runtime type information for one class. Every class derived from `Core::RefCounted` should use the macros `__DeclareClass` and `__ImplementClass` to properly initialize the runtime type information for the class. This will also automatically register the class with the `Core::Factory` object to implement object construction from class name string or fourcc code.

(C) 2006 RadonLabs GmbH
Public Types

typedef RefCounted *(*) Creator ()

define a creation callback function prototype
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rtti</strong> (const char *className, <strong>Util::FourCC</strong> fcc, <strong>Creator</strong> creatorFunc, const <strong>Core::Rtti</strong> *parentClass, SizeT instSize)</td>
<td>constructor</td>
</tr>
<tr>
<td><strong>Rtti</strong> (const char *className, <strong>Creator</strong> creatorFunc, const <strong>Core::Rtti</strong> *parentClass, SizeT instSize)</td>
<td>legacy constructor without <strong>FourCC</strong> for Mangalore compatibility</td>
</tr>
<tr>
<td>bool <strong>operator==</strong> (const <strong>Rtti</strong> &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool <strong>operator!=</strong> (const <strong>Rtti</strong> &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; GetName () const</td>
<td>get class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong> GetFourCC () const</td>
<td>get four character code of class</td>
</tr>
<tr>
<td>const <strong>Core::Rtti</strong> * GetParent () const</td>
<td>get pointer to parent class</td>
</tr>
<tr>
<td>SizeT GetInstanceSize () const</td>
<td>get instance size in bytes</td>
</tr>
<tr>
<td><strong>RefCounted</strong> * Create () const</td>
<td>create an object of this class</td>
</tr>
<tr>
<td>bool <strong>IsDerivedFrom</strong> (const <strong>Rtti</strong> &amp;other) const</td>
<td>return true if this rtti is equal or derived from to other rtti</td>
</tr>
<tr>
<td>bool <strong>IsDerivedFrom</strong> (const <strong>Util::String</strong> &amp;otherClassName) const</td>
<td>return true if this rtti is equal or derived from to other rtti, by string</td>
</tr>
<tr>
<td>bool <strong>IsDerivedFrom</strong> (const <strong>Util::FourCC</strong> &amp;otherClassFourCC) const</td>
<td>return true if this rtti is equal or derived from to other rtti, by fourcc</td>
</tr>
<tr>
<td>void <strong>AllocInstanceMemory</strong> ()</td>
<td>allocate instance memory block (called by class new operator)</td>
</tr>
<tr>
<td>void <strong>FreInstanceMemory</strong> (void *ptr)</td>
<td>free instance memory block (called by class delete operator)</td>
</tr>
</tbody>
</table>
Core::Singleton
Core::Singleton Class Reference

#include <singleton.h>
Detailed Description

Implements a system specific **Singleton**

(C) 2007 Radon Labs GmbH
Core::SysFunc
Core::SysFunc Class Reference

#include <sysfunc.h>
Detailed Description

Wrap some platform specific low-level functions.

(C) 2007 Radon Labs GmbH
CoreAnimation::AnimClip
CoreAnimation::AnimClip Class Reference

#include <animclip.h>
Detailed Description

An animation clip is a collection of related animation curves (for instance all curves required to animate a character).

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AnimClip ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>SetName (const Util::StringAtom &amp;n)</strong></td>
<td>set the name of the clip</td>
</tr>
<tr>
<td><strong>GetName () const</strong></td>
<td>get the name of the clip</td>
</tr>
<tr>
<td><strong>SetNumKeys (SizeT numKeys)</strong></td>
<td>set the number of keys per animation curve in the clip</td>
</tr>
<tr>
<td><strong>GetNumKeys () const</strong></td>
<td>get the number of keys per animation curve in the clip</td>
</tr>
<tr>
<td><strong>SetKeyStride (SizeT stride)</strong></td>
<td>set the key stride (number of float's between keys of the same curve)</td>
</tr>
<tr>
<td><strong>GetKeyStride () const</strong></td>
<td>get the key stride</td>
</tr>
<tr>
<td><strong>SetKeyDuration (Timing::Tick d)</strong></td>
<td>set the duration of a key in ticks</td>
</tr>
<tr>
<td><strong>GetKeyDuration () const</strong></td>
<td>get the duration of a key</td>
</tr>
<tr>
<td><strong>GetClipDuration () const</strong></td>
<td>get the clip duration in ticks</td>
</tr>
<tr>
<td><strong>SetPreInfinityType (InfinityType::Code preInfinityType)</strong></td>
<td>set the pre-infinity type</td>
</tr>
<tr>
<td><strong>GetPreInfinityType () const</strong></td>
<td>get the pre-infinity type</td>
</tr>
<tr>
<td><strong>SetPostInfinityType (InfinityType::Code postInfinityType)</strong></td>
<td>set the post-infinity type</td>
</tr>
<tr>
<td><strong>GetPostInfinityType () const</strong></td>
<td>get the post-infinity type</td>
</tr>
<tr>
<td><strong>SetStartKeyIndex (IndexT keyIndex)</strong></td>
<td>set start key index (actual start time of the curve)</td>
</tr>
<tr>
<td>Function Type</td>
<td>Function Name</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>IndexT</td>
<td><strong>GetStartKeyIndex</strong> () const</td>
</tr>
<tr>
<td>Timing::Tick</td>
<td><strong>GetStartTime</strong> () const</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetNumCurves</strong> (SizeT numCurves)</td>
</tr>
<tr>
<td>SizeT</td>
<td><strong>GetNumCurves</strong> () const</td>
</tr>
<tr>
<td>AnimCurve &amp;</td>
<td><strong>CurveByIndex</strong> (IndexT curveIndex)</td>
</tr>
<tr>
<td>void</td>
<td><strong>BeginEvents</strong> (SizeT numEvents)</td>
</tr>
<tr>
<td>void</td>
<td><strong>AddEvent</strong> (const AnimEvent &amp;event)</td>
</tr>
<tr>
<td>void</td>
<td><strong>EndEvents</strong> ()</td>
</tr>
<tr>
<td>SizeT</td>
<td><strong>GetNumEvents</strong> () const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>HasEvent</strong> (const Util::StringAtom &amp;name)</td>
</tr>
<tr>
<td>const AnimEvent &amp;</td>
<td><strong>GetEventByName</strong> (const Util::StringAtom &amp;name)</td>
</tr>
<tr>
<td>const AnimEvent &amp;</td>
<td><strong>GetEventByIndex</strong> (IndexT i)</td>
</tr>
<tr>
<td>SizeT</td>
<td><strong>GetEventsInRange</strong> (Timing::Tick startTime, Timing::Tick endTime, IndexT &amp;outStartEventIndex)</td>
</tr>
<tr>
<td>void</td>
<td><strong>PrecomputeKeySliceValues</strong> ()</td>
</tr>
<tr>
<td>bool</td>
<td><strong>AreKeySliceValuesValid</strong> () const</td>
</tr>
<tr>
<td>IndexT</td>
<td><strong>GetKeySliceFirstKeyIndex</strong> () const</td>
</tr>
<tr>
<td>SizeT</td>
<td><strong>GetKeySliceByteSize</strong> () const</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get byte size of a key slice in the clip</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

SizeT
CoreAnimation::AnimClip::GetEventsInRange ( Timing::Tick startTime, Timing::Tick endTime, IndexT & outStartEventIndex ) const

get events in time range (return number of events and start event index)

Get events in a specific time range. Return the number of events in the time range, and the index of the start event. This does a linear search on the event array.

void
CoreAnimation::AnimClip::PrecomputeKeySliceValues ( )

pre-compute the key range values

Precompute the 2 key slice values (first key index and key slice size). A key slice is the memory range of all curve-keys at a given key index. The numbers must be pre-computed because only non-static curves have keys in the key-slice.

IndexT
CoreAnimation::AnimClip::GetKeySliceFirstKeyIdx ( ) const [inline]

get index of first key in clip's key range (this is the key index of the first non-static curve)

NOTE: if all curves in the clip are static, then this method will return InvalidIndex, this is not an error situation!

SizeT
CoreAnimation::AnimClip::GetKeySliceByteSize ( ) const [inline]

get byte size of a key slice in the clip

NOTE: if all curves in the clip are static, then this method will return 0,
this is not an error situation!
CoreAnimation::AnimCurve
CoreAnimation::AnimCurve Class Reference

#include <animcurve.h>
Detailed Description

An animation curve describes a set of animation keys in an AnimKeyBuffer. AnimCurves are always part of an AnimClip object, and share properties with all other AnimCurves in their AnimClip object. An AnimCurve may be collapsed into a single key, so that AnimCurves where all keys are identical don't take up any space in the animation key buffer. For performance reasons, AnimCurve's are not as flexible as their Maya counterparts, for instance it is not possible to set the pre- and post-infinity types per curve, but only per clip.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AnimCurve ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>SetActive (bool b)</strong></td>
<td>activate/deactivate the curve, only active curves will be evaluated</td>
</tr>
<tr>
<td><strong>IsActive () const</strong></td>
<td>return true if the animation curve is active</td>
</tr>
<tr>
<td><strong>SetStatic (bool b)</strong></td>
<td>set/clear the static flag</td>
</tr>
<tr>
<td><strong>IsStatic () const</strong></td>
<td>return true if the curve is static</td>
</tr>
<tr>
<td><strong>SetStaticKey (const Math::float4 &amp;staticKey)</strong></td>
<td>set the static key of the curve</td>
</tr>
<tr>
<td><strong>GetStaticKey () const</strong></td>
<td>get the static key of the curve</td>
</tr>
<tr>
<td><strong>SetFirstKeyIndex (IndexT index)</strong></td>
<td>set index of the first key in the AnimKeyBuffer</td>
</tr>
<tr>
<td><strong>GetFirstKeyIndex () const</strong></td>
<td>get index of the first key in the AnimKeyBuffer</td>
</tr>
<tr>
<td><strong>SetCurveType (CurveType::Code curveType)</strong></td>
<td>set the curve type</td>
</tr>
<tr>
<td><strong>GetCurveType () const</strong></td>
<td>get the curve type</td>
</tr>
</tbody>
</table>
CoreAnimation::AnimEvent
CoreAnimation::AnimEvent Class Reference

#include <animevent.h>
Detailed Description

An animation event associates a name with a point in time. The event will be triggered when the play-cursor passes the point in time of the event. AnimEvents are attached to anim clips.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AnimEvent()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>AnimEvent(const Util::StringAtom &amp;name, Timing::Tick time)</code></td>
<td>constructor with name and time</td>
</tr>
<tr>
<td><code>AnimEvent(const Util::StringAtom &amp;name, const Util::StringAtom &amp;category, Timing::Tick time)</code></td>
<td>constructor with name, category and time</td>
</tr>
<tr>
<td><code>void SetName(const Util::StringAtom &amp;name)</code></td>
<td>set the name of the event</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetName() const</code></td>
<td>get the name of the event</td>
</tr>
<tr>
<td><code>void SetCategory(const Util::StringAtom &amp;str)</code></td>
<td>set the name of the category</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetCategory() const</code></td>
<td>get the name of the category</td>
</tr>
<tr>
<td><code>bool HasCategory() const</code></td>
<td>check if has category</td>
</tr>
<tr>
<td><code>void SetTime(Timing::Tick t)</code></td>
<td>set the point-in-time when the event should trigger in seconds</td>
</tr>
<tr>
<td><code>Timing::Tick GetTime() const</code></td>
<td>get the time when the event should trigger</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>operator==</code></td>
<td>(const <code>AnimEvent</code> &amp;a, const <code>AnimEvent</code> &amp;b) equality operator (time only)</td>
</tr>
<tr>
<td><code>operator!=</code></td>
<td>(const <code>AnimEvent</code> &amp;a, const <code>AnimEvent</code> &amp;b) inequality operator (time only)</td>
</tr>
<tr>
<td><code>operator&lt;</code></td>
<td>(const <code>AnimEvent</code> &amp;a, const <code>AnimEvent</code> &amp;b) less-than operator (time only)</td>
</tr>
<tr>
<td><code>operator&gt;</code></td>
<td>(const <code>AnimEvent</code> &amp;a, const <code>AnimEvent</code> &amp;b) greater-than operator (time only)</td>
</tr>
<tr>
<td><code>operator&lt;=</code></td>
<td>(const <code>AnimEvent</code> &amp;a, const <code>AnimEvent</code> &amp;b) less-or-equal operator (time only)</td>
</tr>
<tr>
<td><code>operator&gt;=</code></td>
<td>(const <code>AnimEvent</code> &amp;a, const <code>AnimEvent</code> &amp;b) greater-or-equal operator (time only)</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

CoreAnimation::AnimEventEmitter
CoreAnimation::AnimEventEmitter
Class Reference

#include <animeventemitter.h>
Detailed Description

The **AnimEventEmitter** collects all animevents which are active in the given time range.

This emitter does NOT consider if the clip is used animation driven or not. It always calculates the whole animation duration, not one key-decremented like the play clip job in case of animation driven usage!

(C) 2009 Radon Labs GmbH
| static Util::Array< AnimEvent > | **EmitAnimEvents** (const AnimClip &clip, Timing::Tick start, Timing::Tick end, bool isInfinite) |
Member Function Documentation

```cpp
Util::Array< AnimEvent >
CoreAnimation::AnimEventEmitter::EmitAnimEvents
    ( const AnimClip & clip,
      Timing::Tick start,
      Timing::Tick end,
      bool isInfinite
    ) [static]
```

collects all animevents from given clip
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

CoreAnimation::AnimKeyBuffer
CoreAnimation::AnimKeyBuffer Class Reference

#include <animkeybuffer.h>

Inheritance diagram for CoreAnimation::AnimKeyBuffer:

```
#include <animkeybuffer.h>

Inheritance diagram for CoreAnimation::AnimKeyBuffer:

Core::RefCounted

CoreAnimation::AnimKeyBuffer
```
Detailed Description

A simple buffer of float4 animation keys.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AnimKeyBuffer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~AnimKeyBuffer ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void Setup (SizeT numKeys)</td>
<td>setup the buffer</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>discard the buffer</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>return true if the object has been setup</td>
</tr>
<tr>
<td>SizeT GetNumKeys () const</td>
<td>get number of keys in buffer</td>
</tr>
<tr>
<td>SizeT GetByteSize () const</td>
<td>get buffer size in bytes</td>
</tr>
<tr>
<td>void * Map ()</td>
<td>(obsolete) map key buffer for CPU access</td>
</tr>
<tr>
<td>void Unmap ()</td>
<td>(obsolete) unmap the resource</td>
</tr>
<tr>
<td>bool IsMapped () const</td>
<td>return true if the key buffer is currently mapped</td>
</tr>
<tr>
<td>Math::float4 * GetKeyBufferPointer () const</td>
<td>get direct pointer to key buffer</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>Func</td>
<td>Signature</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td><code>(const Util::FourCC &amp;classFourCC) const</code></td>
</tr>
<tr>
<td><code>IsA</code></td>
<td><code>(const Rtti &amp;rtti) const</code></td>
</tr>
<tr>
<td><code>IsA</code></td>
<td><code>(const Util::String &amp;rttiName) const</code></td>
</tr>
<tr>
<td><code>IsA</code></td>
<td><code>(const Util::FourCC &amp;rttiFourCC) const</code></td>
</tr>
<tr>
<td><code>GetClassName</code></td>
<td>() const</td>
</tr>
<tr>
<td><code>GetClassFourCC</code></td>
<td>() const</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```
int Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.
```

```
void Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.
```

```
void Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```

```
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.
```

```
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
```

```
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
CoreAnimation::AnimResource
CoreAnimation::AnimResource Class Reference

#include <animresource.h>

Inheritance diagram for CoreAnimation::AnimResource:

[Diagram showing the inheritance relationship between Core::RefCounted, Resources::Resource, and CoreAnimation::AnimResource]
Detailed Description

A AnimResource is a collection of related animation clips (for instance all animation clips of a character). AnimResources contain read-only data and are usually shared between several clients. One AnimResource usually contains the data of one animation resource file.

(C) 2008 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>resource states <em>(DO NOT CHANGE ORDER!)</em></td>
</tr>
</tbody>
</table>
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AnimResource ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~AnimResource ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <code>Unload ()</code></td>
<td>unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td><code>SizeT GetNumClips () const</code></td>
<td>get number of animation clips</td>
</tr>
<tr>
<td>const <code>AnimClip &amp; GetClipByIndex (IndexT clipIndex) const</code></td>
<td>get animation clip at index</td>
</tr>
<tr>
<td>bool <code>HasClip (const Util::StringAtom &amp;clipName) const</code></td>
<td>return true if clip exists by name</td>
</tr>
<tr>
<td><code>IndexT GetClipIndexByName (const Util::StringAtom &amp;clipName) const</code></td>
<td>get clip index by name, returns invalid index if not found</td>
</tr>
<tr>
<td>const <code>AnimClip &amp; GetClipByName (const Util::StringAtom &amp;clipName) const</code></td>
<td>get clip by name</td>
</tr>
<tr>
<td>const <code>Ptr&lt; AnimKeyBuffer &gt; &amp; GetKeyBuffer () const</code></td>
<td>get pointer to AnimKeyBuffer</td>
</tr>
<tr>
<td><code>Math::float4 * ComputeKeySlicePointerAndSize (IndexT clipIndex, IndexT keyIndex, SizeT &amp;outSliceByteSize) const</code></td>
<td>get pointer to start of a key slice, and return size of a key slice</td>
</tr>
<tr>
<td>void <code>SetAsyncEnabled (bool b)</code></td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td>bool <code>IsAsyncEnabled () const</code></td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td>void <code>Lock ()</code></td>
<td>set locked to true</td>
</tr>
<tr>
<td>void <code>Unlock ()</code></td>
<td></td>
</tr>
</tbody>
</table>
set locked to false

bool IsLocked () const
  returns true if resource will be used as source for copy process soon

void SetResourceId (const ResourceId &id)
  set the resource identifier

const ResourceId & GetResourceId () const
  get the resource identifier

void SetLoader (const Ptr<ResourceLoader> &loader)
  set optional resource loader

const Ptr<ResourceLoader> & GetLoader () const
  get optional resource loader

void SetSaver (const Ptr<ResourceSaver> &saver)
  set optional resource saver

const Ptr<ResourceSaver> & GetSaver () const
  get optional resource saver

SizeT GetUseCount () const
  get current use count

virtual State Load ()
  load the resource

void SetState (State s)
  set current state (usually only called during Load()!)

State GetState () const
  get current state

bool IsLoaded () const
  return true if current state is Loaded

bool IsPending () const
  return true if current state is Pending

bool LoadFailed () const
  return true if current state is Failed

virtual bool Save ()
  save the resource

int GetRefCount () const
  get the current refcount

void AddRef ()
increment refcount by one

void **Release** ()
*decrement refcount and destroy object if refcount is zero*

bool **IsInstanceOf** (const Rtti &rtti) const
*return true if this object is instance of given class*

bool **IsInstanceOf** (const Util::String &className) const
*return true if this object is instance of given class by string*

bool **IsInstanceOf** (const Util::FourCC &classFourCC) const
*return true if this object is instance of given class by fourcc*

bool **IsA** (const Rtti &rtti) const
*return true if this object is instance of given class, or a derived class*

bool **IsA** (const Util::String &rttiName) const
*return true if this object is instance of given class, or a derived class, by string*

bool **IsA** (const Util::FourCC &rttiFourCC) const
*return true if this object is instance of given class, or a derived class, by fourcc*

const Util::String & **GetClassName** () const
*get the class name*

Util::FourCC **GetClassFourCC** () const
*get the class FourCC code*
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void IncrUseCount()</code></td>
<td>increment use count</td>
</tr>
<tr>
<td><code>void DecrUseCount()</code></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

void
CoreAnimation::AnimResource::Unload() [virtual]

unload the resource, or cancel the pending load

This method is called by the resource subsystem when the resource
should unload its data.

Reimplemented from Resources::Resource.

const float4 *
CoreAnimation::AnimResource::ComputeKeySlicePointerAndSize(IndexT clipIndex,
IndexT keyIndex,
SizeT & outSliceByteSize
) const

get pointer to start of a key slice, and return size of a key slice

Compute start pointer and size of a key slice of a clip in the anim
resource. NOTE: if all anim curves of the clip are static (which means
there are exist no actual keys), then the method will return a NULL
pointer, and an outSliceByteSlice of 0.

Resources::State
Resources::Resource::Load() [virtual, inherited]

load the resource

This loads the resource through the attached resource loader.
Depending on the resource loader, the resource may happen
synchronously or asynchronously. If the resource is loaded
asynchronously, the IsPending() method will return true as long as the
load is in progress, and IsLoaded() will become true when the loading
process has finished. If the load has failed, IsPending() will switch to
false and IsLoaded() will not be true.
save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

`void Core::RefCounted::DumpRefCountingLeaks(); [static, inherited]`

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

CoreAnimation::AnimSampleBuffer
CoreAnimation::AnimSampleBuffer

Class Reference

#include <animsamplebuffer.h>

Inheritance diagram for CoreAnimation::AnimSampleBuffer:
Detailed Description

Stores the result of an animation sampling operation, stores samples key values and sample-counts which keep track of the number of samples which contributed to a mixing result (this is necessary for correct mixing of partial animations).

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AnimSampleBuffer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~AnimSampleBuffer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Setup (const Ptr&lt; AnimResource &gt; &amp;animResource)</strong></td>
<td>setup the object from an animation resource</td>
</tr>
<tr>
<td><strong>void Discard ()</strong></td>
<td>discard the object</td>
</tr>
<tr>
<td><strong>bool IsValid () const</strong></td>
<td>return true if the object has been setup</td>
</tr>
<tr>
<td><strong>SizeT GetNumSamples () const</strong></td>
<td>get the number of samples in the buffer</td>
</tr>
<tr>
<td><strong>Math::float4 * MapSamples ()</strong></td>
<td>(obsolete) gain read/write access to sample buffer</td>
</tr>
<tr>
<td><strong>void UnmapSamples ()</strong></td>
<td>(obsolete) give up access to sample buffer</td>
</tr>
<tr>
<td><strong>uchar * MapSampleCounts ()</strong></td>
<td>(obsolete) gain read/write access to sample counts</td>
</tr>
<tr>
<td><strong>void UnmapSampleCounts ()</strong></td>
<td>(obsolete) give up access to sample counts</td>
</tr>
<tr>
<td><strong>Math::float4 * GetSamplesPointer () const</strong></td>
<td>get direct pointer to samples</td>
</tr>
<tr>
<td><strong>uchar * GetSampleCountsPointer () const</strong></td>
<td>get direct pointer to sample counts</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsInstanceOf</code> (const <code>Util::String</code> &amp;className) const</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><code>IsInstanceOf</code> (const <code>Util::FourCC</code> &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><code>IsA</code> (const <code>Rtti</code> &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><code>IsA</code> (const <code>Util::String</code> &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <code>Util::String</code> &amp;</th>
<th><code>GetClassName</code> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>Util::FourCC</code></th>
<th><code>GetClassFourCC</code> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
CoreAnimation::AnimSampleMixInfo
CoreAnimation::AnimSampleMixInfo
Class Reference

#include <animsamplemixinfo.h>
Detailed Description

A data structure for providing sample/mixing attributes to asynchronous jobs in the CoreAnimation subsystem.

(C) 2009 Radon Labs GmbH
CoreAnimation::AnimUtil
CoreAnimation::AnimUtil Class Reference

#include <animutil.h>
Detailed Description

A class which contains utility methods for animation sampling and mixing.

(C) 2008 Radon Labs GmbH
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sample</strong></td>
<td>(const <em>AnimResource</em> &amp;animResource, IndexT clipIndex, SampleType::Code sampleType, Timing::Tick time, float timeFactor, const <em>AnimSampleBuffer</em> &amp;result)</td>
</tr>
<tr>
<td>OBSOLETE: sample an animation clip at some point in time into an <em>AnimSampleBuffer</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>CreateSampleJob</strong></td>
<td>(const <em>AnimResource</em> &amp;animResource, IndexT clipIndex, SampleType::Code sampleType, Timing::Tick time, float timeFactor, const <em>AnimSampleBuffer</em> &amp;result)</td>
</tr>
<tr>
<td>setup a job object which performs sampling</td>
<td></td>
</tr>
<tr>
<td><strong>CreateSampleAndMixJob</strong></td>
<td>(const <em>AnimResource</em> &amp;animResource, IndexT clipIndex, SampleType::Code sampleType, Timing::Tick time, float timeFactor, float mixWeight, const <em>AnimSampleBuffer</em> &amp;mixIn, const <em>AnimSampleBuffer</em> &amp;result)</td>
</tr>
<tr>
<td>setup a job which performs both sampling and mixing</td>
<td></td>
</tr>
<tr>
<td><strong>ClampKeyIndex</strong></td>
<td>(IndexT keyIndex, const <em>AnimClip</em> &amp;clip)</td>
</tr>
<tr>
<td>clamp key index into valid range</td>
<td></td>
</tr>
<tr>
<td><strong>InbetweenTicks</strong></td>
<td>(Timing::Tick sampleTime, const <em>AnimClip</em> &amp;clip)</td>
</tr>
<tr>
<td>compute inbetween ticks for a given sample time</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

void CoreAnimation::AnimUtil::Sample(const Ptr<AnimResource> &animResource, IndexT clipIndex, SampleType::Code sampleType, Timing::Tick time, float timeFactor, const Ptr<AnimSampleBuffer> &result)
{
    // OBSOLETE: sample an animation clip at some point in time into an AnimSampleBuffer.
    // NOTE: this method is obsolete
    // NOTE: The sampler will *NOT* the start time of the clip into account!
    // TODO: separate delta computation from default sampling, set curveindex from jointname in characterinstance !!!
}

Ptr<Job> CoreAnimation::AnimUtil::CreateSampleJob(const Ptr<AnimResource> &animResource, IndexT clipIndex, SampleType::Code sampleType, Timing::Tick time, float timeFactor, const Ptr<AnimSampleBuffer> &resultBuffer)
{
    // setup a job object which performs sampling
    // Create a job object which is setup to perform simple animation sampling.
}

Ptr<Job> CoreAnimation::AnimUtil::CreateSampleAndMixJob(const Ptr<AnimResource> &animResource, IndexT clipIndex, SampleType::Code sampleType, timing::Tick time, float timeFactor, const Ptr<AnimSampleBuffer> &resultBuffer)
{
    // setup a job object which performs sampling
    // Create a job object which is setup to perform simple animation sampling.
}
Timing::Tick time,
float timeFactor,
float mixWeight,
const Ptr<
AnimSampleBuffer> mixIn,
> &
const Ptr<
AnimSampleBuffer> resultBuffer
> &
)

setup a job which performs both sampling and mixing

Create a job which performs both sampling and mixing.

IndexT
CoreAnimation::AnimUtil::ClampKeyIndex ( IndexT keyIndex,
const 
AnimClip clip
&
)

clamp key index into valid range

Clamp key indices into the valid range, take pre-infinity and post-infinity type into account.

Timing::Tick
CoreAnimation::AnimUtil::InbetweenTicks ( Timing::Tick sampleTime,
const 
AnimClip &
clip
)

compute inbetween ticks for a given sample time

Compute the inbetween-ticks between two frames for a given sample time.
CoreAnimation::CurveType
CoreAnimation::CurveType Class Reference

#include <curvetype.h>
Detailed Description

Describes the general data type of the keys stored in an animation curve.

(C) 2008 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>animation curve types</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static Code</th>
<th>FromString (const Util::String &amp;str)</th>
</tr>
</thead>
<tbody>
<tr>
<td>convert from string</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static Util::String</th>
<th>ToString (Code c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>convert to string</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
CoreAnimation::InfinityType
CoreAnimation::InfinityType Class Reference

#include <infinitytype.h>
Detailed Description

Describes how time position outsides of an animation curve's scope are handled.

(C) 2008 Radon Labs GmbH
<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>anim infinity types</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static Code FromString (const Util::String &amp;str)</code></td>
<td>convert from string</td>
</tr>
<tr>
<td><code>static Util::String ToString (Code c)</code></td>
<td>convert to string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by `doxygen` at Fri Mar 26 15:21:42 2010
CoreAnimation::ManagedAnimation
CoreAnimation::ManagedAnimation
Class Reference

#include <managedanimresource.h>
Detailed Description

Managed wrapper for AnimResource class.

(C) 2008 Radon Labs GmbH
CoreAnimation::Nax2Header
CoreAnimation::Nax2Header Struct Reference

#include <naxfileformatstructs.h>
Detailed Description

legacy NAX2 file format structs
CoreAnimation::Nax3Header
CoreAnimation::Nax3Header Struct Reference

#include <naxfileformatstructs.h>
Detailed Description

NAX3 file format structs.

NOTE: keep all header-structs 4-byte aligned!
CoreAnimation::SampleType
CoreAnimation::SampleType Class Reference

#include <samplertype.h>
Detailed Description

Describes how an animation curve should be sampled.

(C) 2008 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>animation sample types</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FromString</code></td>
<td><code>static Code FromString (const Util::String &amp;str)</code></td>
<td>convert from string</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td><code>static Util::String ToString (Code c)</code></td>
<td>convert to string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](http://www.doxygen.nl) at Fri Mar 26 15:21:42 2010
CoreAnimation::StreamAnimationLoader
CoreAnimation::StreamAnimationLoader
Class Reference

#include <streamanimationloader.h>

Inheritance diagram for CoreAnimation::StreamAnimationLoader:
Detailed Description

Initialize a CoreAnimation::AnimResource from the content of a stream.

(C) 2008 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool CanLoadAsync () const</td>
<td>override in subclass: return true if asynchronous loading is supported (default is true)</td>
</tr>
<tr>
<td>virtual bool OnLoadRequested ()</td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td>virtual void OnLoadCancelled ()</td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td>virtual bool OnPending ()</td>
<td>call frequently while after OnLoadRequested() to put Resource into loaded state</td>
</tr>
<tr>
<td>virtual void OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>virtual void OnRemoveFromResource ()</td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>bool IsAttachedToResource () const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetResource () const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td>Resource::State GetState () const</td>
<td>return current state</td>
</tr>
<tr>
<td>virtual void Reset ()</td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>return true if this object is instance of given class by fourcc</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by string</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by fourcc</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool <code>SetupResourceFromStream</code> (const <code>Ptr&lt; IO::Stream &gt;</code> &amp;stream)</td>
<td>setup the <code>AnimResource</code> object from a stream</td>
</tr>
<tr>
<td>bool <code>SetupFromNax3</code> (const <code>Ptr&lt; IO::Stream &gt;</code> &amp;stream)</td>
<td>setup the <code>AnimResource</code> from an NAX3 stream</td>
</tr>
<tr>
<td>void <code>SetState</code> (Resource::State s)</td>
<td>set current state</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool Resources::StreamResourceLoader::CanLoadAsync() const [virtual, inherited]

override in subclass: return true if asynchronous loading is supported (default is true)

Indicate whether this resource loader supports asynchronous loading. The default is true. Override this method in a subclass and return false otherwise.

Reimplemented from Resources::ResourceLoader.

Reimplemented in Direct3D9::D3D9StreamShaderLoader.

bool Resources::StreamResourceLoader::OnLoadRequested() [virtual, inherited]

called by resource when a load is requested

Handle the generic load request. In the asynchronous case, this method will fire a ReadStream message and go into pending state. The client will then call OnPending() periodically which checks whether the ReadStream message has been handled and continue accordingly. In the synchronous state the method will create an IO::Stream object and call the SetupResourceFromStream() method directly.

Reimplemented from Resources::ResourceLoader.

Reimplemented in Resources::D3D9TextureStreamer.

void Resources::StreamResourceLoader::OnLoadCancelled() [virtual, inherited]

called by resource to cancel a pending load

This method is called when the currently pending asynchronous load
request should be cancelled.

Reimplemented from Resources::ResourceLoader.

```cpp
bool
Resources::StreamResourceLoader::OnPending( ) [virtual, inherited]
```

call frequently while after OnLoadRequested() to put Resource into loaded state

This method is called periodically by the client when the resource is in pending state. The pending ReadStream message will be checked, and if it has been handled successfully the SetupResourceFromStream() method will be called and the Resource will go into Loaded state. If anything goes wrong, the resource will go into Failed state.

Reimplemented from Resources::ResourceLoader.

```cpp
int
Core::RefCounted::GetRefCount( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void
Core::RefCounted::AddRef( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]
```
get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::Adapter
CoreGraphics::Adapter Class Reference

#include <adapter.h>
Detailed Description

Display adapter enum.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enum</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FromString</code></td>
<td><code>static Code FromString (const Util::String &amp;str)</code></td>
<td>convert adapter code from string</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td><code>static Util::String ToString (Code code)</code></td>
<td>convert adapter code to string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
CoreGraphics::AdapterInfo
CoreGraphics::AdapterInfo Class Reference

#include <adapterinfo.h>
Detailed Description

Contains information about a given display adapter. This info can be used to identify a specific piece of hardware or driver version. Use DisplayDevice::GetAdapterInfo() to obtain information about existing display adapters.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AdapterInfo ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>void SetDriverName (const Util::String &amp;s)</strong></td>
<td>Set driver name</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetDriverName ()</strong></td>
<td>Get human readable driver name</td>
</tr>
<tr>
<td><strong>void SetDescription (const Util::String &amp;s)</strong></td>
<td>Set description string</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetDescription ()</strong></td>
<td>Get human readable description</td>
</tr>
<tr>
<td><strong>void SetDeviceName (const Util::String &amp;s)</strong></td>
<td>Set device name</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetDeviceName ()</strong></td>
<td>Get human readable device name</td>
</tr>
<tr>
<td><strong>void SetDriverVersionLowPart (uint v)</strong></td>
<td>Set driver version low part</td>
</tr>
<tr>
<td><strong>uint GetDriverVersionLowPart ()</strong></td>
<td>Get low part of driver version</td>
</tr>
<tr>
<td><strong>void SetDriverVersionHighPart (uint v)</strong></td>
<td>Set driver version high part</td>
</tr>
<tr>
<td><strong>uint GetDriverVersionHighPart ()</strong></td>
<td>Get high part of driver version</td>
</tr>
<tr>
<td><strong>void SetVendorId (uint id)</strong></td>
<td>Set vendor id</td>
</tr>
<tr>
<td><strong>uint GetVendorId ()</strong></td>
<td>Get vendor identifier</td>
</tr>
<tr>
<td><strong>void SetDeviceId (uint id)</strong></td>
<td>Set device id</td>
</tr>
<tr>
<td><strong>uint GetDeviceId ()</strong></td>
<td>Get device identifier</td>
</tr>
<tr>
<td><strong>void SetSubSystemId (uint id)</strong></td>
<td>Set subsystem id</td>
</tr>
<tr>
<td><strong>uint GetSubSystemId ()</strong></td>
<td>Get subsystem identifier</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>void SetRevision (uint r)</td>
<td>set hardware revision</td>
</tr>
<tr>
<td>uint GetRevision () const</td>
<td>get hardware revision identifier</td>
</tr>
<tr>
<td>void SetGuid (const Util::Guid &amp;g)</td>
<td>set driver/chipset pair guid</td>
</tr>
<tr>
<td>const Util::Guid &amp; GetGuid () const</td>
<td>get guid for driver/chipset pair</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

CoreGraphics::AntiAliasQuality
CoreGraphics::AntiAliasQuality Class Reference

#include <antialiasquality.h>
Detailed Description

Anti-alias quality levels.

(C) 2006 Radon Labs GmbH
<table>
<thead>
<tr>
<th>enum</th>
<th><strong>Code</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FromString</code></td>
<td>convert from string</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>convert to string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
CoreGraphics::BatchType
CoreGraphics::BatchType Class Reference

#include <batchtype.h>
Detailed Description

Batch type hints for the render device and render targets. Indicates the type of objects that are rendered in a batch.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
<th>batch type enum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FromString</code></td>
<td>convert from string</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>convert to string</td>
</tr>
</tbody>
</table>

**static BatchType::Code** `FromString` (const `Util::String` &str)

**static `Util::String`** `ToString` (`BatchType::Code` c)
CoreGraphics::DisplayDevice
CoreGraphics::DisplayDevice Class Reference

#include <displaydevice.h>

Inheritance diagram for CoreGraphics::DisplayDevice:
Detailed Description

A `DisplayDevice` object represents the display where the `RenderDevice` presents the rendered frame.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>DisplayDevice</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual</td>
<td><strong>~DisplayDevice</strong> ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>bool</td>
<td><strong>AdapterExists</strong> (CoreGraphics::Adapter::Code adapter)</td>
<td>Check if the adapter actually exists</td>
</tr>
<tr>
<td></td>
<td><strong>GetAvailableDisplayModes</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>SupportsDisplayMode</strong></td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::DisplayMode</td>
<td><strong>GetCurrentAdapterDisplayMode</strong></td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::AdapterInfo</td>
<td><strong>GetAdapterInfo</strong></td>
<td></td>
</tr>
<tr>
<td>virtual bool</td>
<td><strong>Open</strong> ()</td>
<td></td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Close</strong> ()</td>
<td></td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>ProcessWindowMessages</strong> ()</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>HWND GetHwnd () const</td>
<td>get the application window HWND</td>
<td></td>
</tr>
<tr>
<td>void SetAdapter (CoreGraphics::Adapter::Code a)</td>
<td>set display adapter (make sure adapter exists!)</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::Adapter::Code GetAdapter () const</td>
<td>get display adapter</td>
<td></td>
</tr>
<tr>
<td>void SetDisplayMode (const CoreGraphics::DisplayMode &amp;m)</td>
<td>set display mode (make sure the display mode is supported!)</td>
<td></td>
</tr>
<tr>
<td>const CoreGraphics::DisplayMode &amp; GetDisplayMode () const</td>
<td>get display mode</td>
<td></td>
</tr>
<tr>
<td>void SetAntiAliasQuality (CoreGraphics::AntiAliasQuality::Code aa)</td>
<td>set antialias quality</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::AntiAliasQuality::Code GetAntiAliasQuality () const</td>
<td>get antialias quality</td>
<td></td>
</tr>
<tr>
<td>void SetFullscreen (bool b)</td>
<td>set windowed/fullscreen mode</td>
<td></td>
</tr>
<tr>
<td>bool IsFullscreen () const</td>
<td>get windowed/fullscreen mode</td>
<td></td>
</tr>
<tr>
<td>void SetDisplayModeSwitchEnabled (bool b)</td>
<td>enable display mode switch when running fullscreen (default is true);</td>
<td></td>
</tr>
<tr>
<td>bool IsDisplayModeSwitchEnabled () const</td>
<td>is display mode switch enabled for fullscreen?</td>
<td></td>
</tr>
<tr>
<td>void SetTripleBufferingEnabled (bool b)</td>
<td>enable triple buffer for fullscreen (default is double buffering)</td>
<td></td>
</tr>
<tr>
<td>bool IsTripleBufferingEnabled () const</td>
<td>is triple buffer enabled for fullscreen?</td>
<td></td>
</tr>
<tr>
<td>void SetAlwaysOnTop (bool b)</td>
<td>set always-on-top behaviour</td>
<td></td>
</tr>
<tr>
<td>bool IsAlwaysOnTop () const</td>
<td>get always-on-top behaviour</td>
<td></td>
</tr>
<tr>
<td>void SetVerticalSyncEnabled (bool b)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>bool IsVerticalSyncEnabled</code></td>
<td>get vertical sync flag</td>
<td></td>
</tr>
<tr>
<td><code>void SetIconName (const Util::String &amp;s)</code></td>
<td>set optional window icon resource name</td>
<td></td>
</tr>
<tr>
<td><code>const Util::String &amp; GetIconName</code></td>
<td>get optional window icon resource name</td>
<td></td>
</tr>
<tr>
<td><code>void SetParentWindow (void *h)</code></td>
<td>set optional parent window handle</td>
<td></td>
</tr>
<tr>
<td><code>void * GetParentWindow</code></td>
<td>get optional parent window handle</td>
<td></td>
</tr>
<tr>
<td><code>void SetWindowTitle (const Util::String &amp;t)</code></td>
<td>set window title string (can be changed anytime)</td>
<td></td>
</tr>
<tr>
<td><code>const Util::String &amp; GetWindowTitle</code></td>
<td>get window title string</td>
<td></td>
</tr>
<tr>
<td><code>bool IsOpen</code></td>
<td>return true if display is currently open</td>
<td></td>
</tr>
<tr>
<td><code>void AttachEventHandler (const Ptr&lt;CoreGraphics::DisplayEventHandler &amp;h&gt;)</code></td>
<td>attach a display event handler</td>
<td></td>
</tr>
<tr>
<td><code>void RemoveEventHandler (const Ptr&lt;CoreGraphics::DisplayEventHandler &amp;h&gt;)</code></td>
<td>remove a display event handler</td>
<td></td>
</tr>
<tr>
<td><code>int GetRefCount</code></td>
<td>get the current refcount</td>
<td></td>
</tr>
<tr>
<td><code>void AddRef</code></td>
<td>increment refcount by one</td>
<td></td>
</tr>
<tr>
<td><code>void Release</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
<td></td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
<td></td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>(const <code>Util::FourCC &amp;classFourCC</code>) const</td>
<td></td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
<td></td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>(const <code>Rtti &amp;rtti</code>) const</td>
<td></td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, derived class</td>
<td></td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>(const <code>Util::String &amp;rttiName</code>) const</td>
<td></td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, derived class, by string</td>
<td></td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>(const <code>Util::FourCC &amp;rttiFourCC</code>) const</td>
<td></td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, derived class, by fourcc</td>
<td></td>
</tr>
<tr>
<td><code>GetClassName</code></td>
<td>() const</td>
<td></td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td><code>GetClassFourCC</code></td>
<td>() const</td>
<td></td>
</tr>
<tr>
<td></td>
<td>get the class <code>FourCC</code> code</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool</td>
<td><code>OpenWindow</code></td>
<td>open the application window</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>CloseWindow</code></td>
<td>close the application window</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnMinimized</code></td>
<td>called on WM_SIZE when window is minimized</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnRestored</code></td>
<td>called on WM_SIZE when window is restored</td>
</tr>
<tr>
<td>virtual bool</td>
<td><code>OnSetCursor</code></td>
<td>called on WM_SETCURSOR</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnPaint</code></td>
<td>called on WM_PAINT</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnSetFocus</code></td>
<td>called on WM_SETFOCUS</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnKillFocus</code></td>
<td>called on WM_KILLFOCUS</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnCloseRequested</code></td>
<td>called on WM_CLOSE to request if window should be closed</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnToggleFullscreenWindowed</code></td>
<td>called when Alt-Enter is pressed</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnKeyDown</code></td>
<td>called on WM_KEYDOWN</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnKeyUp</code></td>
<td>called on WM_KEYUP</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnChar</code></td>
<td>called on WM_CHAR</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnMouseButton</code></td>
<td>called on mouse button event</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnMouseMove</code></td>
<td>called on WM_MOUSEMOVE</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnMouseWheel</code></td>
<td>called on WM_MOUSEWHEEL</td>
</tr>
<tr>
<td>virtual bool</td>
<td><code>Input::Key::Code</code></td>
<td><code>TranslateKeyCode</code> (WPARAM wParam) const</td>
</tr>
</tbody>
</table>
translate a Windows virtual key code into a Nebula3 key code

**Math::float2**

**ComputeAbsMousePos** (LPARAM lParam) const

*compute absolute mouse position from lParam*

**Math::float2**

**ComputeNormMousePos** (const Math::float2 &absMousePos) const

*compute normalized mouse position from absolute mouse pos*

**bool**

**NotifyEventHandlers** (const CoreGraphics::DisplayEvent &e)

*notify event handlers about an event*
## Static Protected Member Functions

<table>
<thead>
<tr>
<th>static LRESULT CALLBACK</th>
<th>WinProc (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)</th>
</tr>
</thead>
<tbody>
<tr>
<td>the WinProc</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool Direct3D9::D3D9DisplayDevice::AdapterExists(CoreGraphics::Adapter::Code adapter) [inherited]
```

check if the adapter actually exists

This method checks if the given adapter actually exists.

Reimplemented from `Base::DisplayDeviceBase`.

```cpp
Util::Array<DisplayMode> Direct3D9::D3D9DisplayDevice::GetAvailableDisplayModes(CoreGraphics::Adapter::Code CoreGraphics::PixelFormat::Code)
```

get available display modes on given adapter

Enumerate the available display modes on the given adapter in the given pixel format. If the adapter doesn't exist on this machine, an empty array is returned.

Reimplemented from `Base::DisplayDeviceBase`.

```cpp
void Win32::Win32DisplayDevice::ProcessWindowMessages()
```

process window system messages, call this method once per frame

Polls for and processes window messages. Call this message once per frame in your render loop. If the user clicks the window close button, or hits Alt-F4, a CloseRequested input event will be sent.

Reimplemented from `Base::DisplayDeviceBase`.

```cpp
LRESULT CALLBACK Win32::Win32DisplayDevice::WinProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
```

[static, protected,]
The Nebula3 WinProc:

```cpp
bool Win32::Win32DisplayDevice::OpenWindow() [protected, virtual, inherited]
open the application window
Open the application window.
```

```cpp
void Win32::Win32DisplayDevice::CloseWindow() [protected, virtual, inherited]
close the application window
Close the application window.
```

```cpp
Input::Key::Code Win32::Win32DisplayDevice::TranslateKeyCode(WPARAM wParam) const [protected, inherited]
translate a Windows virtual key code into a Nebula3 key code
Helper method which translates a Win32 virtual key code into a Nebula key code.
```

```cpp
void Base::DisplayDeviceBase::SetWindowTitle(const Util::String& str) [inherited]
set window title string (can be changed anytime)
Set the window title. An application should be able to change the window title at any time, that's why this is a virtual method, so that a subclass may override it.
```

```cpp
void Base::DisplayDeviceBase::AttachEventHandler(const Ptr/CoreGraphics::DisplayEventHandler& h) [inherited]
attach a display event handler
```
Attach an event handler to the display device.

```cpp
void Base::DisplayDeviceBase::RemoveEventHandler ( const CoreGraphics::DisplayEventHandler & h ) [inherited]
```

remove a display event handler

Remove an event handler from the display device.

```cpp
bool Base::DisplayDeviceBase::NotifyEventHandlers ( const CoreGraphics::DisplayEvent & e ) [protected, inherited]
```

notify event handlers about an event

Notify all event handlers about an event.

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

get the class name
Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::DisplayEvent
CoreGraphics::DisplayEvent Class Reference

#include <displayevent.h>
**Detailed Description**

Display events are sent by the display device to registered display event handlers.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>event codes</td>
</tr>
</tbody>
</table>
Public Member Functions

<table>
<thead>
<tr>
<th>DisplayEvent (Code c)</th>
<th>constructor with event code</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisplayEvent (Code c, const Math::float2 &amp;absPos, const Math::float2 &amp;normPos)</td>
<td>constructor with event code and mouse pos</td>
</tr>
<tr>
<td>DisplayEvent (Code c, Input::Key::Code k)</td>
<td>constructor with key code</td>
</tr>
<tr>
<td>DisplayEvent (Code c, Input::Char chr)</td>
<td>constructor with character</td>
</tr>
<tr>
<td>DisplayEvent (Code c, Input::MouseButton::Code b, const Math::float2 &amp;absPos, const Math::float2 &amp;normPos)</td>
<td>constructor with mouse button and mouse pos</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>GetEventCode () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Math::float2 &amp;</td>
<td>GetAbsMousePos () const</td>
</tr>
<tr>
<td>const Math::float2 &amp;</td>
<td>GetNormMousePos () const</td>
</tr>
<tr>
<td>Input::Key::Code</td>
<td>GetKey () const</td>
</tr>
<tr>
<td>Input::Char</td>
<td>GetChar () const</td>
</tr>
<tr>
<td>Input::MouseButton::Code</td>
<td>GetMouseButton () const</td>
</tr>
</tbody>
</table>

get event code

get absolute mouse pos (in pixels)

get normalized mouse pos (from 0.0 to 1.0)

get key code

get character code

get mouse button code

The Nebula Device 3 documentation generated by doxygen at Fri Mar
CoreGraphics::DisplayEventHandler
CoreGraphics::DisplayEventHandler
Class Reference

#include <displayeventhandler.h>

Inheritance diagram for CoreGraphics::DisplayEventHandler:

- Core::RefCounted
- CoreGraphics::DisplayEventHandler
- CoreGraphics::ThreadSafeDisplayEventHandler
- Win32::Win32InputDisplayEventHandler
Detailed Description

A display event handler object is notified by the `DisplayDevice` about noteworthy window events, for instance when the mouse is moved, the window is minimized, and so on. To get notified about those events, derive a class from `DisplayEventHandler` and attach to the display device via `DisplayDevice::AttachEventHandler()`.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DisplayEventHandler ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~DisplayEventHandler ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>OnAttach ()</strong></td>
<td>Called when the event handler is attached to the DisplayDevice</td>
</tr>
<tr>
<td>virtual void <strong>OnRemove ()</strong></td>
<td>Called when the event handler is removed from the DisplayDevice</td>
</tr>
<tr>
<td>virtual bool <strong>PutEvent</strong> (const DisplayEvent &amp;event)</td>
<td>Called by DisplayDevice when an event happens</td>
</tr>
<tr>
<td><strong>GetRefCount ()</strong> const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetClassName</strong> () const</td>
<td>Get the class name</td>
</tr>
</tbody>
</table>
Util::FourCC GetClassFourCC () const

get the class FourCC code
<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>virtual bool</th>
<th><strong>HandleEvent</strong> (const <strong>DisplayEvent</strong> &amp;event)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>called when an event should be processed, override this method in your subclass</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool CoreGraphics::DisplayEventHandler::PutEvent (const DisplayEvent & e ) [virtual]

called by DisplayDevice when an event happens

This method is called by the DisplayDevice when an event happens. The default behaviour of this class is to call the HandleEvent() method directly. Subclasses of DisplayEventHandler may choose to implement a different behaviour.

Reimplemented in CoreGraphics::ThreadSafeDisplayEventHandler.

bool CoreGraphics::DisplayEventHandler::HandleEvent (const DisplayEvent & e ) [protected, virtual]

called when an event should be processed, override this method in your subclass

Handle a display event. This method is usually called by PutEvent(), but subclasses of DisplayEventHandler may choose to implement a different behaviour. Override this method in your subclass to process the incoming event.

Reimplemented in CoreGraphics::ThreadSafeDisplayEventHandler, and Win32::Win32InputDisplayEventHandler.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.
Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::DisplayMode
CoreGraphics::DisplayMode Class Reference

#include <displaymode.h>
Detailed Description

Describe a fullscreen display mode or window dimensions.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DisplayMode()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>DisplayMode(uint x, uint y, uint w, uint h)</code></td>
<td>windowed mode constructor</td>
</tr>
<tr>
<td><code>DisplayMode(uint w, uint h, PixelFormat::Code p)</code></td>
<td>fullscreen constructor</td>
</tr>
<tr>
<td><code>DisplayMode(uint x, uint y, uint w, uint h, PixelFormat::Code p)</code></td>
<td>generic constructor</td>
</tr>
<tr>
<td><code>bool operator==(const DisplayMode &amp;rhs)</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const DisplayMode &amp;rhs)</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>void SetXPos(uint x)</code></td>
<td>set x position</td>
</tr>
<tr>
<td><code>uint GetXPos()</code></td>
<td>get x position</td>
</tr>
<tr>
<td><code>void SetYPos(uint y)</code></td>
<td>set y position</td>
</tr>
<tr>
<td><code>uint GetYPos()</code></td>
<td>get y position</td>
</tr>
<tr>
<td><code>void SetWidth(uint w)</code></td>
<td>set width</td>
</tr>
<tr>
<td><code>uint GetWidth()</code></td>
<td>get width</td>
</tr>
<tr>
<td><code>void SetHeight(uint h)</code></td>
<td>set height</td>
</tr>
<tr>
<td><code>uint GetHeight()</code></td>
<td>get height</td>
</tr>
<tr>
<td><code>void SetPixelFormat(PixelFormat::Code p)</code></td>
<td>set pixel format</td>
</tr>
<tr>
<td><code>PixelFormat::Code GetPixelFormat()</code></td>
<td>get pixel format</td>
</tr>
</tbody>
</table>
void SetAspectRatio (float a)

set aspect ratio

float GetAspectRatio () const

get aspect ratio
Constructor & Destructor Documentation

CoreGraphics::DisplayMode::DisplayMode ( uint x, 
    uint y, 
    uint w, 
    uint h 
) [inline]

windowed mode constructor

This constructor is suitable for windowed modes.

CoreGraphics::DisplayMode::DisplayMode ( uint w, 
    uint h, 
    PixelFormat::Code p 
) [inline]

fullscreen constructor

This constructor is suitable for fullscreen modes.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
CoreGraphics::ImageFileFormat
CoreGraphics::ImageFileFormat Class Reference

#include <imagefileformat.h>
Detailed Description

Image file formats supported by StreamTextureSaver.

(C) 2007 Radon Labs GmbH
# Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>image file formats</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FromString</code> (const <code>Util::String</code> &amp;str)</td>
<td>convert from string</td>
</tr>
<tr>
<td><code>ToString</code> (Code c)</td>
<td>convert to string</td>
</tr>
<tr>
<td><code>FromMediaType</code> (const <code>IO::MediaType</code> &amp;mediaType)</td>
<td>convert from media type (MIME)</td>
</tr>
<tr>
<td><code>ToMediaType</code> (Code c)</td>
<td>convert to media type (MIME)</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:42 2010
CoreGraphics::IndexBuffer
CoreGraphics::IndexBuffer Class Reference

#include <indexbuffer.h>

Inheritance diagram for CoreGraphics::IndexBuffer:
Detailed Description

A resource which holds an array of indices into an array of vertices.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>Usage</strong></th>
<th>resource usage flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><strong>State</strong></td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>Unload</strong> ()</td>
<td>unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td>void * <strong>Map</strong> (MapType mapType)</td>
<td>map index buffer for CPU access</td>
</tr>
<tr>
<td>void <strong>Unmap</strong> ()</td>
<td>unmap the resource</td>
</tr>
<tr>
<td>void <strong>SetD3D9IndexBuffer</strong> (IDirect3DIndexBuffer9 *ptr)</td>
<td>set d3d9 index buffer pointer</td>
</tr>
<tr>
<td>IDirect3DIndexBuffer9 * <strong>GetD3D9IndexBuffer</strong> () const</td>
<td>get d3d9 index buffer pointer</td>
</tr>
<tr>
<td>void <strong>SetIndexType</strong> (CoreGraphics::IndexType::Code type)</td>
<td>set the index type (Index16 or Index32)</td>
</tr>
<tr>
<td>CoreGraphics::IndexType::Code <strong>GetIndexType</strong> () const</td>
<td>get the index type (Index16 or Index32)</td>
</tr>
<tr>
<td>void <strong>SetNumIndices</strong> (SizeT num)</td>
<td>set number of indices</td>
</tr>
<tr>
<td>SizeT <strong>GetNumIndices</strong> () const</td>
<td>get number of indices</td>
</tr>
<tr>
<td>void <strong>SetUsage</strong> (Usage usage)</td>
<td>set resource usage type</td>
</tr>
<tr>
<td>Usage <strong>GetUsage</strong> () const</td>
<td>get resource usage type</td>
</tr>
<tr>
<td>void <strong>SetAccess</strong> (Access access)</td>
<td>set resource cpu access type</td>
</tr>
<tr>
<td>Access <strong>GetAccess</strong> () const</td>
<td>get cpu access type</td>
</tr>
<tr>
<td>void <strong>SetAsyncEnabled</strong> (bool b)</td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td>bool <strong>IsAsyncEnabled</strong> () const</td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>void Lock()</td>
<td>set locked to true</td>
</tr>
<tr>
<td>void Unlock()</td>
<td>set locked to false</td>
</tr>
<tr>
<td>bool IsLocked()</td>
<td>returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td>void SetResourceId(const Resourceld &amp;id)</td>
<td>set the resource identifier</td>
</tr>
<tr>
<td>const Resourceld &amp; GetResourceId() const</td>
<td>get the resource identifier</td>
</tr>
<tr>
<td>void SetLoader(const Ptr&lt;ResourceLoader&gt; &amp;loader)</td>
<td>set optional resource loader</td>
</tr>
<tr>
<td>const Ptr&lt;ResourceLoader&gt; &amp; GetLoader() const</td>
<td>get optional resource loader</td>
</tr>
<tr>
<td>void SetSaver(const Ptr&lt;ResourceSaver&gt; &amp;saver)</td>
<td>set optional resource saver</td>
</tr>
<tr>
<td>const Ptr&lt;ResourceSaver&gt; &amp; GetSaver() const</td>
<td>get optional resource saver</td>
</tr>
<tr>
<td>SizeT GetUseCount() const</td>
<td>get current use count</td>
</tr>
<tr>
<td>virtual State Load()</td>
<td>load the resource</td>
</tr>
<tr>
<td>void SetState(State s)</td>
<td>set current state (usually only called during Load())</td>
</tr>
<tr>
<td>State GetState() const</td>
<td>get current state</td>
</tr>
<tr>
<td>bool IsLoaded() const</td>
<td>return true if current state is Loaded</td>
</tr>
<tr>
<td>bool IsPending() const</td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td>bool LoadFailed() const</td>
<td>return true if current state is Failed</td>
</tr>
<tr>
<td>virtual bool Save()</td>
<td>save the resource</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool Isa (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool Isa (const Util::String &amp;rttiName) const</td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool Isa (const Util::FourCC &amp;rttiFourCC) const</td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>Get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <strong>IncrUseCount</strong> ()</td>
<td>increment use count</td>
</tr>
<tr>
<td>void <strong>DecrUseCount</strong> ()</td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

Resource::State
Resources::Resource::Load( ) [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the IsPending() method will return true as long as the load is in progress, and IsLoaded() will become true when the loading process has finished. If the load has failed, IsPending() will switch to false and IsLoaded() will not be true.

bool
Resources::Resource::Save( ) [virtual, inherited]

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

int
Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::IndexType
CoreGraphics::IndexType Class Reference

#include <indextype.h>

------------------------------------------
Detailed Description

Data type of vertex indices (16 bit or 32 bit).

(C) 2006 Radon Labs GmbH
# Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td></td>
</tr>
</tbody>
</table>

*index types enum*
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SizeOf (IndexType::Code type)</code></td>
<td>get byte size of index</td>
</tr>
<tr>
<td><code>ToString (IndexType::Code type)</code></td>
<td>convert index type to string</td>
</tr>
<tr>
<td><code>FromString (const Util::String &amp;str)</code></td>
<td>convert string to index type</td>
</tr>
</tbody>
</table>
CoreGraphics::MemoryIndexBufferLoader
CoreGraphics::MemoryIndexBufferLoader
Class Reference

#include <memoryindexbufferloader.h>

Inheritance diagram for CoreGraphics::MemoryIndexBufferLoader:

- Core::RefCounted
- Resources::ResourceLoader
- Base::MemoryIndexBufferLoaderBase
- Win360::D3DMemoryIndexBufferLoader
- CoreGraphics::MemoryIndexBufferLoader
Detailed Description

Initialize an index buffer object from index data in memory.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool <code>OnLoadRequested()</code></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td>void <code>Setup(CoreGraphics::IndexType::Code indexType, SizeT numIndices, void *indexDataPtr, SizeT indexDataSize);</code></td>
<td>setup index buffer from existing data, or provide 0 pointer if empty index buffer should be created</td>
</tr>
<tr>
<td>virtual void <code>OnAttachToResource(const Ptr&lt;Resource&gt; &amp;res)</code></td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>virtual void <code>OnRemoveFromResource()</code></td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>bool <code>IsAttachedToResource()</code> const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt;Resource&gt; &amp; <code>GetResource()</code> const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td>virtual bool <code>CanLoadAsync()</code> const</td>
<td>return true if asynchronous loading is supported</td>
</tr>
<tr>
<td>virtual void <code>OnLoadCancelled()</code></td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td>virtual bool <code>OnPending()</code></td>
<td>call frequently while after <code>OnLoadRequested()</code> to put Resource into loaded state</td>
</tr>
<tr>
<td>const Resource::State <code>GetState()</code> const</td>
<td>return current state</td>
</tr>
<tr>
<td>virtual void <code>Reset()</code></td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td>int <code>GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef()</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>increment refcount by one</code></td>
<td></td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
void SetState (Resource::State S)

set current state
```
Member Function Documentation

bool
Win360::D3D9MemoryIndexBufferLoader::OnLoadRequested() [virtual, inherited]
called by resource when a load is requested

This will create a D3D9 IndexBuffer using the data provided by our Setup() method and set our resource object (which must be a D3D9IndexBuffer object).

Reimplemented from Resources::ResourceLoader.

bool
Resources::ResourceLoader::CanLoadAsync() const [virtual, inherited]
return true if asynchronous loading is supported

This method should be overridden in a subclass and indicates whether the resource loader supports asynchronous resource loading. If asynchronous loading is requested, the OnLoadRequested() method will return immediately and the Resource object will be put into Pending state. Afterwards, the Resource object needs to poll the ResourceLoader using the OnPending method, which will eventually setup the Resource object.

Reimplemented in Direct3D9::D3D9StreamShaderLoader, Models::StreamModelLoader, and Resources::StreamResourceLoader.

void
Resources::ResourceLoader::OnLoadCancelled() [virtual, inherited]
called by resource to cancel a pending load

This method is called by our Resource object if a pending asynchronous load should be cancelled.

Reimplemented in Models::StreamModelLoader, and
bool Resources::ResourceLoader::OnPending() [virtual, inherited]
call frequently while after OnLoadRequested() to put Resource into loaded state

This method should be called at some time after OnLoadRequested() as long as the ResourceLoader is in the Pending state. This will check whether the asynchronous loader job has finished, and if yes, setup the Resource object, bringing it from the Pending into the Loaded state. If something goes wrong, the ResourceLoader will go into the Failed state. If the outstanding loader job isn't finished yet, the ResourceLoader should remain in Pending state, and the method should return false. Otherwise the Resource should be initialized, and the method should return true.

Reimplemented in Models::StreamModelLoader, and Resources::StreamResourceLoader.

int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.


```
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

---

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
CoreGraphics::MemoryMeshLoader
CoreGraphics::MemoryMeshLoader
Class Reference

#include <memorymeshloader.h>

Inheritance diagram for CoreGraphics::MemoryMeshLoader:
Detailed Description

Setup a **Mesh** object from a given vertex, index buffer and primitive group.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemoryMeshLoader</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>void SetUsage (Base::ResourceBase::Usage usage)</code></td>
<td>set the intended resource usage (default is UsageImmutable)</td>
</tr>
<tr>
<td><strong>Base::ResourceBase::Usage</strong> GetUsage () const</td>
<td>get resource usage</td>
</tr>
<tr>
<td><strong>void SetAccess (Base::ResourceBase::Access access)</strong></td>
<td>set the intended resource access (default is AccessNone)</td>
</tr>
<tr>
<td><strong>Base::ResourceBase::Access</strong> GetAccess () const</td>
<td>get the resource access</td>
</tr>
<tr>
<td><code>void SetVertexBuffer (const CoreGraphics::VertexBuffer &amp;vBuffer)</code></td>
<td>set vertex buffer</td>
</tr>
<tr>
<td><code>void SetIndexBuffer (const CoreGraphics::IndexBuffer &amp;iBuffer)</code></td>
<td>set index buffer</td>
</tr>
<tr>
<td><code>void SetPrimitiveGroups (const Util::Array&lt;CoreGraphics::PrimitiveGroup&gt; &amp;pGroup)</code></td>
<td>set primitive group</td>
</tr>
<tr>
<td><strong>virtual bool OnLoadRequested ()</strong></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td><strong>virtual void OnAttachToResource (const Resource &amp;res)</strong></td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td><strong>virtual void OnRemoveFromResource ()</strong></td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td><strong>bool IsAttachedToResource ()</strong> const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td><code>const Resource &amp; GetResource () const</code></td>
<td></td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CanLoadAsync() const</td>
<td>Return true if asynchronous loading is supported</td>
</tr>
<tr>
<td>OnLoadCancelled()</td>
<td>Called by resource to cancel a pending load</td>
</tr>
<tr>
<td>OnPending()</td>
<td>Call frequently while after OnLoadRequested() to put Resource into loaded state</td>
</tr>
<tr>
<td>GetState() const</td>
<td>Return current state</td>
</tr>
<tr>
<td>Reset()</td>
<td>Resets loader-stats e.g. state</td>
</tr>
<tr>
<td>GetRefCount() const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>AddRef()</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>Release()</td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>IsInstanceOf(const Util::String &amp;className) const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>IsInstanceOf(const Util::FourCC &amp;classFourCC) const</td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>IsA (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>IsA (const Util::String &amp;rttiName) const</td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>GetClassName() const</td>
<td>Get the class name</td>
</tr>
<tr>
<td>GetClassFourCC() const</td>
<td>Get the class fourcc</td>
</tr>
</tbody>
</table>
get the class FourCC code
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>SetupMeshFromMemory()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>setup mesh resource from given memory data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>SetState(Resource::State s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set current state</td>
</tr>
</tbody>
</table>
bool Resources::ResourceLoader::CanLoadAsync ( ) const [virtual, inherited]

return true if asynchronous loading is supported

This method should be overridden in a subclass and indicates whether the resource loader supports asynchronous resource loading. If asynchronous loading is requested, the OnLoadRequested() method will return immediately and the Resource object will be put into Pending state. Afterwards, the Resource object needs to poll the ResourceLoader using the OnPending method, which will eventually setup the Resource object.

Reimplemented in Direct3D9::D3D9StreamShaderLoader, Models::StreamModelLoader, and Resources::StreamResourceLoader.

void Resources::ResourceLoader::OnLoadCancelled ( ) [virtual, inherited]

called by resource to cancel a pending load

This method is called by our Resource object if a pending asynchronous load should be cancelled.

Reimplemented in Models::StreamModelLoader, and Resources::StreamResourceLoader.

bool Resources::ResourceLoader::OnPending ( ) [virtual, inherited]

call frequently while after OnLoadRequested() to put Resource into loaded state

This method should be called at some time after OnLoadRequested() as long as the ResourceLoader is in the Pending state. This will check whether the asynchronous loader job has finished, and if yes,
setup the **Resource** object, bringing it from the Pending into the Loaded state. If something goes wrong, the **ResourceLoader** will go into the Failed state. If the outstanding loader job isn't finished yet, the **ResourceLoader** should remain in Pending state, and the method should return false. Otherwise the **Resource** should be initialized, and the method should return true.

Reimplemented in **Models::StreamModelLoader**, and **Resources::StreamResourceLoader**.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code
Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
CoreGraphics::MemoryVertexBufferLoader
CoreGraphics::MemoryVertexBufferLoader

Class Reference

#include <memoryvertexbufferloader.h>

Inheritance diagram for CoreGraphics::MemoryVertexBufferLoader:

```
CoreRefCounted

Resources::ResourceLoader

Base::MemoryVertexBufferLoaderBase

Win360::D3D9MemoryVertexBufferLoader

CoreGraphics::MemoryVertexBufferLoader
```
Detailed Description

Initialize a vertex buffer object from vertex data in memory.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual bool OnLoadRequested()</code></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td><code>void Setup (const Util::Array&lt; CoreGraphics::VertexComponent &gt; &amp;vertexComponents, SizeT numVertices, void *vertexDataPtr, SizeT vertexDataSize, CoreGraphics::VertexBuffer::Usage usage)</code></td>
<td>setup vertex buffer data, must remain valid until <code>OnLoadRequested()</code> is called!</td>
</tr>
<tr>
<td><code>virtual void OnAttachToResource (const Ptr&lt;Resource&gt; &amp;res)</code></td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td><code>virtual void OnRemoveFromResource ()</code></td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td><code>bool IsAttachedToResource () const</code></td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td><code>const Ptr&lt;Resource&gt; &amp; GetResource () const</code></td>
<td>get pointer to resource</td>
</tr>
<tr>
<td><code>virtual bool CanLoadAsync () const</code></td>
<td>return true if asynchronous loading is supported</td>
</tr>
<tr>
<td><code>virtual void OnLoadCancelled ()</code></td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td><code>virtual bool OnPending ()</code></td>
<td>call frequently while after <code>OnLoadRequested()</code> to put <code>Resource</code> into loaded state</td>
</tr>
<tr>
<td><code>Resource::State GetState () const</code></td>
<td>return current state</td>
</tr>
<tr>
<td><code>virtual void Reset ()</code></td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti)</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName)</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName()</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC()</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

```c
void SetState (Resource::State S)

set current state
```
Member Function Documentation

```cpp
bool Win360::D3D9MemoryVertexBufferLoader::OnLoadRequested() [virtual, inherited]
called by resource when a load is requested

This will create a D3D9 vertex buffer and vertex declaration object from the data provided in the Setup() method and setup our resource object (which must be a D3D9VertexBuffer object).

Reimplemented from Resources::ResourceLoader.
```

```cpp
bool Resources::ResourceLoader::CanLoadAsync() const [virtual, inherited]
return true if asynchronous loading is supported

This method should be overriden in a subclass and indicates whether the resource loader supports asynchronous resource loading. If asynchronous loading is requested, the OnLoadRequested() method will return immediately and the Resource object will be put into Pending state. Afterwards, the Resource object needs to poll the ResourceLoader using the OnPending method, which will eventually setup the Resource object.

Reimplemented in Direct3D9::D3D9StreamShaderLoader, Models::StreamModelLoader, and Resources::StreamResourceLoader.
```

```cpp
void Resources::ResourceLoader::OnLoadCancelled() [virtual, inherited]
called by resource to cancel a pending load

This method is called by our Resource object if a pending asynchronous load should be cancelled.

Reimplemented in Models::StreamModelLoader, and
```
Resources::StreamResourceLoader.

bool Resources::ResourceLoader::OnPending() [virtual, inherited]

call frequently while after OnLoadRequested() to put Resource into loaded state

This method should be called at some time after OnLoadRequested() as long as the ResourceLoader is in the Pending state. This will check whether the asynchronous loader job has finished, and if yes, setup the Resource object, bringing it from the Pending into the Loaded state. If something goes wrong, the ResourceLoader will go into the Failed state. If the outstanding loader job isn't finished yet, the ResourceLoader should remain in Pending state, and the method should return false. Otherwise the Resource should be initialized, and the method should return true.

Reimplemented in Models::StreamModelLoader, and Resources::StreamResourceLoader.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::Mesh
CoreGraphics::Mesh Class Reference

#include <mesh.h>

Inheritance diagram for CoreGraphics::Mesh:

```
Core::RefCounted
  
Resources::Resource
  
Base::MeshBase
  
CoreGraphics::Mesh
```
Detailed Description

A mesh contains a vertex buffer, an optional index buffer and a number of `PrimitiveGroup` objects. Meshes can be loaded directly from a mesh resource file.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>State</th>
</tr>
</thead>
</table>

resource states *(DO NOT CHANGE ORDER!)*
### Public Member Functions

<table>
<thead>
<tr>
<th>virtual void</th>
<th><strong>Unload ( )</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>unload mesh resource</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>HasVertexBuffer ( ) const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>return true if the mesh has a vertex buffer</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetVertexBuffer (const <strong>CoreGraphics::VertexBuffer</strong> &amp;vb)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set the vertex buffer object</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>CoreGraphics::VertexBuffer</strong> &amp;</th>
<th><strong>GetVertexBuffer ( ) const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the vertex buffer object</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>HasIndexBuffer ( ) const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>return true if the mesh has an index buffer</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetIndexBuffer (const <strong>CoreGraphics::IndexBuffer</strong> &amp;ib)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set the index buffer object</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>CoreGraphics::IndexBuffer</strong> &amp;</th>
<th><strong>GetIndexBuffer ( ) const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the index buffer object</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetPrimitiveGroups (const <strong>Util::Array</strong>&lt; <strong>CoreGraphics::PrimitiveGroup</strong> &gt; &amp;groups)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set primitive groups</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SizeT</th>
<th><strong>GetNumPrimitiveGroups ( ) const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the number of primitive groups in the mesh</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>CoreGraphics::PrimitiveGroup</strong> &amp;</th>
<th><strong>GetPrimitiveGroupAtIndex (IndexT i) const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get primitive group at index</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>ApplyPrimitives (IndexT primGroupIndex)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>apply mesh data for rendering in renderdevice</em></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void SetAsyncEnabled (bool b)</code></td>
<td>Request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td><code>bool IsAsyncEnabled () const</code></td>
<td>Return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>void Lock ()</code></td>
<td>Set locked to true</td>
</tr>
<tr>
<td><code>void Unlock ()</code></td>
<td>Set locked to false</td>
</tr>
<tr>
<td><code>bool IsLocked () const</code></td>
<td>Returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td><code>void SetResourceId (const ResourceId &amp;id)</code></td>
<td>Set the resource identifier</td>
</tr>
<tr>
<td><code>const ResourceId &amp; GetResourceId () const</code></td>
<td>Get the resource identifier</td>
</tr>
<tr>
<td><code>void SetLoader (const Ptr&lt;ResourceLoader&gt; &amp;loader)</code></td>
<td>Set optional resource loader</td>
</tr>
<tr>
<td><code>const Ptr&lt;ResourceLoader&gt; &amp; GetLoader () const</code></td>
<td>Get optional resource loader</td>
</tr>
<tr>
<td><code>void SetSaver (const Ptr&lt;ResourceSaver&gt; &amp;saver)</code></td>
<td>Set optional resource saver</td>
</tr>
<tr>
<td><code>const Ptr&lt;ResourceSaver&gt; &amp; GetSaver () const</code></td>
<td>Get optional resource saver</td>
</tr>
<tr>
<td><code>SizeT GetUseCount () const</code></td>
<td>Get current use count</td>
</tr>
<tr>
<td><code>virtual State Load ()</code></td>
<td>Load the resource</td>
</tr>
<tr>
<td><code>void SetState (State s)</code></td>
<td>Set current state (usually only called during <code>Load()</code>)</td>
</tr>
<tr>
<td><code>State GetState () const</code></td>
<td>Get current state</td>
</tr>
<tr>
<td><code>bool IsLoaded () const</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool <code>IsPending</code> () const</td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td>bool <code>LoadFailed</code> () const</td>
<td>return true if current state is Failed</td>
</tr>
<tr>
<td>virtual bool <code>Save</code> ()</td>
<td>save the resource</td>
</tr>
<tr>
<td>int <code>GetRefCount</code> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef</code> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release</code> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Util::String</code> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Util::FourCC</code> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Util::String</code> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp;</td>
<td><code>GetClassName</code> () const</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------</td>
</tr>
</tbody>
</table>

*get the class FourCC code*
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><strong>IncrUseCount</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>increment use count</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>DecrUseCount</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

**Resource::State**

Resources::Resource::Load( ) [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the `IsPending()` method will return true as long as the load is in progress, and `IsLoaded()` will become true when the loading process has finished. If the load has failed, `IsPending()` will switch to false and `IsLoaded()` will not be true.

```cpp
bool Resources::Resource::Save( ) [virtual, inherited]
```

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

```cpp
int Core::RefCounted::GetRefCount( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release( ) [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```
const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::MousePointer
CoreGraphics::MousePointer Class Reference

#include <mousepointer.h>
Detailed Description

Contains information how to render a mouse pointer through the CoreGraphics::MouseRenderer class. Please note not all parameters are supported on all platforms (like the orientation vector).

The pointer works in screen space coordinates, where (0,0) is the middle of the screen, and -1 is top/left.

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MousePointer ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>void SetResourceId (const Resources::ResourceId &amp;texResId)</code></td>
<td>set mouse pointer texture resource id</td>
</tr>
<tr>
<td><code>const Resources::ResourceId &amp; GetResourceId () const</code></td>
<td>get mouse pointer texture resource id</td>
</tr>
<tr>
<td><code>void SetPosition (const Math::point &amp;pos)</code></td>
<td>set position of the mouse pointer in screen coordinates</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetPosition () const</code></td>
<td>get position of the mouse pointer in screen coordinates</td>
</tr>
<tr>
<td><code>void SetSize (const Math::vector &amp;size)</code></td>
<td>set size in screen coordinates (screen size is 2.0)</td>
</tr>
<tr>
<td><code>const Math::vector &amp; GetSize () const</code></td>
<td>get size in screen coordinates</td>
</tr>
<tr>
<td><code>void SetOrientation (const Math::vector &amp;orient)</code></td>
<td>set orientation of the mouse pointer as horizon vector (1,0) is horizontal</td>
</tr>
<tr>
<td><code>const Math::vector &amp; GetOrientation () const</code></td>
<td>get orientation of the mouse pointer as horizon vector</td>
</tr>
<tr>
<td><code>void SetHotspot (const Math::point &amp;hotSpot)</code></td>
<td>set hot spot of the mouse vector (-1,-1 is top/left, 1,1 is bottom/right)</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetHotspot () const</code></td>
<td>get hot spot of the mouse vector</td>
</tr>
<tr>
<td><code>void SetAlpha (Math::scalar alpha)</code></td>
<td>set alpha value</td>
</tr>
<tr>
<td><code>Math::scalar GetAlpha () const</code></td>
<td>get hot spot of the mouse vector</td>
</tr>
</tbody>
</table>
The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
CoreGraphics::MouseRenderDevice
CoreGraphics::MouseRenderDevice
Class Reference

#include <mouserenderdevice.h>

Inheritance diagram for CoreGraphics::MouseRenderDevice:

```
   Core::RefCounted
    \     / \
   /     \  
Base::MouseRenderDeviceBase
     |     |
    /     \
  CoreGraphics::MouseRenderDevice
```
Detailed Description

Platform-wrapper for mouse pointer rendering.

(C) 2009 Radon Labs GmbH
Public Member Functions

MouseRenderDevice ()
constructor

virtual ~MouseRenderDevice ()
destructor

void Setup ()
open the mouse renderer

void Discard ()
close the mouse renderer

bool IsValid () const
return true if the mouse renderer is valid

void PreloadTextures (const Util::Array<Resources::ResourceId> &texResIds)
load mouse pointer textures

void UpdatePointers (const Util::Array<CoreGraphics::MousePointer> &pointers)
update the mouse renderer

void RenderPointers ()
render mouse pointer(s)

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</th>
<th>return true if this object is instance of given class, or a derived class, by string</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

void Base::MouseRenderDeviceBase::PreloadTextures
   (const Util::Array<Resources::ResourceId> & texResIds) [inherited]

load mouse pointer textures

This method must be called to preload texture used by the mouse renderer. The method may be called at any time (also several times).

void Base::MouseRenderDeviceBase::UpdatePointers
   (const Util::Array<CoreGraphics::MousePointer> & pointers) [inline, inherited]

update the mouse renderer

Update mouse pointers for rendering in the current frame. On some platforms, more then one mouse pointer exists, so this method takes an array of MousePointer objects. Calling this method will replace the previous array of MousePoiners.

void Base::MouseRenderDeviceBase::RenderPointers
   ( ) [inherited]

render mouse pointer(s)

This method should render the pointers describes by the last call to UpdatePointers(). Override this method in a derived platform- specific class.

int Core::RefCounted::GetRefCount
   ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef
   ( ) [inline, inherited]
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
CoreGraphics::MultipleRenderTarget
CoreGraphics::MultipleRenderTarget
Class Reference

#include <multiplerendertarget.h>

Inheritance diagram for CoreGraphics::MultipleRenderTarget:
Detailed Description

A multiple render targets wraps up to 4 rendertargets into a C++ object. The special default render target represents the backbuffer and default depth/stencil surface.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void AddRenderTarget (const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp;rt)</code></td>
<td>set rendertarget</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; GetRenderTarget (IndexT i) const</code></td>
<td>get rendertarget at index</td>
</tr>
<tr>
<td><code>SizeT GetNumRendertargets () const</code></td>
<td>get number of rendertargets used</td>
</tr>
<tr>
<td><code>void BeginPass ()</code></td>
<td>begin rendering to the render target</td>
</tr>
<tr>
<td><code>void BeginBatch (CoreGraphics::BatchType::Code batchType)</code></td>
<td>begin a batch</td>
</tr>
<tr>
<td><code>void EndBatch ()</code></td>
<td>end current batch</td>
</tr>
<tr>
<td><code>void EndPass ()</code></td>
<td>end current render pass</td>
</tr>
<tr>
<td><code>void SetClearColor (IndexT i, const Math::float4 &amp;color)</code></td>
<td>set clear color of rendertarget at index</td>
</tr>
<tr>
<td><code>void SetClearDepth (float d)</code></td>
<td>set clear depth</td>
</tr>
<tr>
<td><code>float GetClearDepth () const</code></td>
<td>get clear depth</td>
</tr>
<tr>
<td><code>void SetClearStencil (uchar s)</code></td>
<td>set clear stencil value</td>
</tr>
<tr>
<td><code>uchar GetClearStencil () const</code></td>
<td>get clear stencil value</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA (const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DumpRefCountingLeaks()</code></td>
<td><code>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</code></td>
</tr>
</tbody>
</table>
### Static Public Attributes

<table>
<thead>
<tr>
<th>static const IndexT</th>
<th>MaxNumRenderTargets</th>
<th>= 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>max number of rendertargets</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

`int Core::RefCounted::GetRefCount ( ) const [inline, inherited]`

get the current refcount

Return the current refcount of the object.

`void Core::RefCounted::AddRef ( ) [inline, inherited]`

increment refcount by one

Increment the refcount of the object.

`void Core::RefCounted::Release ( ) [inline, inherited]`

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

`const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]`

get the class name

Get the class name of the object.

`Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]`

get the class FourCC code

Get the class FourCC of the object.

`void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]`

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
This method should be called as the very last before an application exits.
CoreGraphics::PixelFormat
CoreGraphics::PixelFormat Class Reference

#include <pixelformat.h>
Detailed Description

Pixel format enumeration.

FIXME: use DX10 notations (more flexible but less readable...)

(C) 2006 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>enums</em></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static Code</th>
<th><strong>FromString</strong> (const Util::String &amp;str)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>convert from string</em></td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>ToString</strong> (Code code)</td>
</tr>
<tr>
<td></td>
<td><em>convert to string</em></td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
PixelFormat::Code CoreGraphics::PixelFormat::FromUtilString(const Util::String & str) [static]
```

**convert from string**

Convert a pixel format string into a pixel format code.

```cpp
Util::String CoreGraphics::PixelFormat::ToUtilString(PixelFormat::Code code) [static]
```

**convert to string**

Convert pixel format code into a string.
CoreGraphics::PrimitiveGroup
CoreGraphics::PrimitiveGroup Class Reference

#include <primitivegroup.h>
Detailed Description

Defines a group of primitives as a subset of a vertex buffer and index buffer plus the primitive topology (triangle list, etc...).

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PrimitiveGroup()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>void SetBaseVertex(IndexT i)</code></td>
<td>Set base vertex index</td>
</tr>
<tr>
<td><code>IndexT GetBaseVertex()</code></td>
<td>Get index of base vertex</td>
</tr>
<tr>
<td><code>void SetNumVertices(SizeT n)</code></td>
<td>Set number of vertices</td>
</tr>
<tr>
<td><code>SizeT GetNumVertices()</code></td>
<td>Get number of vertices</td>
</tr>
<tr>
<td><code>void SetBaseIndex(IndexT i)</code></td>
<td>Set base index index</td>
</tr>
<tr>
<td><code>IndexT GetBaseIndex()</code></td>
<td>Get base index index</td>
</tr>
<tr>
<td><code>void SetNumIndices(SizeT n)</code></td>
<td>Set number of indices</td>
</tr>
<tr>
<td><code>SizeT GetNumIndices()</code></td>
<td>Get number of indices</td>
</tr>
<tr>
<td><code>void SetPrimitiveTopology(PrimitiveTopology::Code topology)</code></td>
<td>Set the primitive topology</td>
</tr>
<tr>
<td><code>PrimitiveTopology::Code GetPrimitiveTopology()</code></td>
<td>Get the primitive topology</td>
</tr>
<tr>
<td><code>void SetBoundingBox(const Math::bbox &amp;b)</code></td>
<td>Set the primitive group's local bounding box</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetBoundingBox()</code></td>
<td>Get the primitive group's local bounding box</td>
</tr>
<tr>
<td><code>SizeT GetNumPrimitives()</code></td>
<td>Get computed number of primitives</td>
</tr>
</tbody>
</table>
CoreGraphics::PrimitiveTopology
CoreGraphics::PrimitiveTopology Class Reference

#include <primitivetopology.h>
Detailed Description

The primitive topology for a draw call.

(C) 2006 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enumeration</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

| static Code FromString (const Util::String &str) | convert from string |
| static Util::String ToString (Code code) | convert to string |
| static SizeT NumberOfVertices (Code topology, SizeT numPrimitives) | compute number of vertices/indices given a primitive topology and number of primitives |
| static SizeT NumberOfPrimitives (Code topology, SizeT numVertices) | compute number of primitives given a primitive type and number of vertices/indices |
Member Function Documentation

SizeT CoreGraphics::PrimitiveTopology::NumberOfVertices (Code topology,
SizeT numPrimitives)
[static]

compute number of vertices/indices given a primitive topology and number of primitives

Computes the number of required vertices for a given primitive topology and number of primitives.

SizeT CoreGraphics::PrimitiveTopology::NumberOfPrimitives (Code topology,
SizeT numVertices)
[static]

compute number of primitives given a primitive type and number of vertices/indices

Computes the number of primitives from a given primitive topology and number of vertices (the opposite of ComputeNumberOfVertices()).
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

CoreGraphics::RenderDevice
CoreGraphics::RenderDevice Class Reference

#include <renderdevice.h>

Inheritance diagram for CoreGraphics::RenderDevice:

```
Core::RefCounted
   |
   V
Base::RenderDeviceBase
   |
   V
Direct3D::D3D9RenderDevice
   |
   V
CoreGraphics::RenderDevice
```
Detailed Description

The central rendering object of the Nebula3 core graphics system. This is basically an encapsulation of the Direct3D device. The render device will presents its backbuffer to the display managed by the `CoreGraphics::DisplayDevice` singleton.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RenderDevice ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~RenderDevice ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>IDirect3DDevice9 * <strong>GetDirect3DDevice ()</strong></td>
<td>const get pointer to the d3d device</td>
</tr>
<tr>
<td><strong>Open ()</strong></td>
<td>open the device</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>close the device</td>
</tr>
<tr>
<td><strong>BeginFrame ()</strong></td>
<td>begin complete frame</td>
</tr>
<tr>
<td>void <strong>SetStreamSource</strong> (IndexT streamIndex, const Ptr<a href="">CoreGraphics::VertexBuffer</a> &amp;vb, IndexT offsetVertexIndex)</td>
<td>set the current vertex stream source</td>
</tr>
<tr>
<td>void <strong>SetVertexLayout</strong> (const Ptr<a href="">CoreGraphics::VertexLayout</a> &amp;vl)</td>
<td>set current vertex layout</td>
</tr>
<tr>
<td>void <strong>SetIndexBuffer</strong> (const Ptr<a href="">CoreGraphics::IndexBuffer</a> &amp;ib)</td>
<td>set current index buffer</td>
</tr>
<tr>
<td>void <strong>Draw ()</strong></td>
<td>draw current primitives</td>
</tr>
<tr>
<td>void <strong>DrawIndexedInstanced</strong> (SizeT numInstances)</td>
<td>draw indexed, instanced primitives (see method header for details)</td>
</tr>
<tr>
<td>void <strong>EndPass ()</strong></td>
<td>end current pass</td>
</tr>
<tr>
<td>void <strong>EndFrame ()</strong></td>
<td>end complete frame</td>
</tr>
<tr>
<td>void <strong>Present ()</strong></td>
<td>present the rendered scene</td>
</tr>
<tr>
<td>Function/Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SaveScreenshot</td>
<td>Save a screenshot to the provided stream</td>
</tr>
<tr>
<td>GetPresentParams</td>
<td>Get the present parameters</td>
</tr>
<tr>
<td>SetOverrideDefaultRenderTarget</td>
<td>Set an override for the default render target (call before Open!)</td>
</tr>
<tr>
<td>IsOpen</td>
<td>Return true if currently open</td>
</tr>
<tr>
<td>AttachEventHandler</td>
<td>Attach a render event handler</td>
</tr>
<tr>
<td>RemoveEventHandler</td>
<td>Remove a render event handler</td>
</tr>
<tr>
<td>GetDefaultRenderTarget</td>
<td>Get default render target</td>
</tr>
<tr>
<td>HasPassRenderTarget</td>
<td>Is a pass render target set?</td>
</tr>
<tr>
<td>GetPassRenderTarget</td>
<td>Get current set pass render target</td>
</tr>
<tr>
<td>BeginPass</td>
<td>Begin rendering a frame pass</td>
</tr>
<tr>
<td>BeginPass</td>
<td>Begin rendering a frame pass with a multiple...</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void BeginBatch(CoreGraphics::BatchType::Code batchType, const Ptr&lt;CoreGraphics::ShaderInstance&gt; &amp;batchShader)</code></td>
<td><code>begin rendering a batch inside</code></td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::VertexBuffer &gt; &amp; GetStreamVertexBuffer (IndexT streamIndex) const</code></td>
<td><code>get currently set vertex buffer</code></td>
</tr>
<tr>
<td><code>IndexT GetStreamVertexOffset (IndexT streamIndex) const</code></td>
<td><code>get currently set vertex stream offset</code></td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::VertexLayout &gt; &amp; GetVertexLayout () const</code></td>
<td><code>get current vertex layout</code></td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::IndexBuffer &gt; &amp; GetIndexBuffer () const</code></td>
<td><code>get current index buffer</code></td>
</tr>
<tr>
<td><code>void SetPrimitiveGroup (const CoreGraphics::PrimitiveGroup &amp;p)</code></td>
<td><code>set current primitive group</code></td>
</tr>
<tr>
<td><code>const CoreGraphics::PrimitiveGroup &amp; GetPrimitiveGroup () const</code></td>
<td><code>get current primitive group</code></td>
</tr>
<tr>
<td><code>void EndBatch ()</code></td>
<td><code>end current batch</code></td>
</tr>
<tr>
<td><code>bool IsInBeginFrame () const</code></td>
<td><code>check if inside BeginFrame</code></td>
</tr>
<tr>
<td><code>bool GetVisualizeMipMaps () const</code></td>
<td><code>get visualization of mipmaps flag</code></td>
</tr>
<tr>
<td><code>void SetVisualizeMipMaps (bool val)</code></td>
<td><code>set visualization of mipmaps flag</code></td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td><code>get the current refcount</code></td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td><code>increment refcount by one</code></td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td><code>decrement refcount and destroy object if refcount zero</code></td>
</tr>
</tbody>
</table>
bool IsInstanceOf (const Rtti &rtti) const
    return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
    return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
    return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
    return true if this object is instance of given class, derived class

bool IsA (const Util::String &rttiName) const
    return true if this object is instance of given class, derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
    return true if this object is instance of given class, derived class, by fourcc

const Util::String & GetClassName () const
    get the class name

Util::FourCC GetClassFourCC () const
    get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static bool</td>
<td><strong>CanCreate</strong> ()</td>
<td>test if a compatible render device can be created on this machine</td>
</tr>
<tr>
<td>static IDirect3D *</td>
<td><strong>GetDirect3D</strong> ()</td>
<td>get pointer to Direct3D interface, opens Direct3D if not happened yet</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
**Static Public Attributes**

<table>
<thead>
<tr>
<th>static const IndexT</th>
<th>MaxNumVertexStreams</th>
<th>= 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>max number of vertex streams</td>
<td></td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
bool NotifyEventHandlers (const CoreGraphics::RenderEvent &e)

notify event handlers about an event
```
Member Function Documentation

```cpp
bool Direct3D9::D3D9RenderDevice::CanCreate( ) [static, inherited]
```

test if a compatible render device can be created on this machine

Test if the right Direct3D version is installed by trying to open Direct3D.

Reimplemented from `Base::RenderDeviceBase`.

```cpp
IDirect3D * Direct3D9::D3D9RenderDevice::GetDirect3D( ) [static, inherited]
```

get pointer to Direct3D interface, opens Direct3D if not happened yet

Get a pointer to the Direct3D interface. Opens Direct3D if not happened yet.

```cpp
IDirect3DDevice9 * Direct3D9::D3D9RenderDevice::GetDirect3DDevice() const [inherited]
```

get pointer to the d3d device

Return a pointer to d3d device. Asserts that the device exists.

```cpp
bool Direct3D9::D3D9RenderDevice::Open( ) [inherited]
```

open the device

Open the render device. When successful, the `RenderEvent::DeviceOpen` will be sent to all registered event handlers after the Direct3D device has been opened.

Reimplemented from `Base::RenderDeviceBase`.

```cpp
void Direct3D9::D3D9RenderDevice::Close( ) [inherited]
```
close the device

Close the render device. The RenderEvent::DeviceClose will be sent to all registered event handlers.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
bool Direct3D9::D3D9RenderDevice::BeginFrame() [inherited]
```

begin complete frame

Begin a complete frame. Call this once per frame before any rendering happens. If rendering is not possible for some reason (e.g. a lost device) the method will return false. This method may also send the DeviceLost and DeviceRestored RenderEvents to attached event handlers.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
void Direct3D9::D3D9RenderDevice::SetStreamSource(IndexT streamIndex, const Ptr<CoreGraphics::VertexBuffer> &vb, IndexT offsetVertexInc)
```

set the current vertex stream source

Sets the vertex buffer to use for the next **Draw()**.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
void Direct3D9::D3D9RenderDevice::SetVertexLayout(const Ptr<CoreGraphics::VertexLayout> &vl) [inherited]
```

set current vertex layout

Sets the vertex layout for the next **Draw()**
set current index buffer

Sets the index buffer to use for the next Draw().

draw current primitives

Draw the current primitive group. Requires a vertex buffer, an optional index buffer and a primitive group to be set through the respective methods. To use non-indexed rendering, set the number of indices in the primitive group to 0.

draw indexed, instanced primitives (see method header for details)

Draw N instances of the current primitive group. Requires the following setup:

- vertex stream 0: vertex buffer with instancing data, one vertex per instance
- vertex stream 1: vertex buffer with instance geometry data
- index buffer: index buffer for geometry data
- primitive group: the primitive group which describes one instance
- vertex declaration: describes a combined vertex from stream 0 and stream 1
void Direct3D9::D3D9RenderDevice::EndPass() [inherited]

direct current pass

End the current rendering pass. This will flush all texture stages in order to keep the d3d9 resource reference counts consistent without too much hassle.

Reimplemented from Base::RenderDeviceBase.

void Direct3D9::D3D9RenderDevice::EndFrame() [inherited]

direct complete frame

End a complete frame. Call this once per frame after rendering has happened and before Present(), and only if BeginFrame() returns true.

Reimplemented from Base::RenderDeviceBase.

void Direct3D9::D3D9RenderDevice::Present() [inherited]

present the rendered scene

NOTE: Present() should be called as late as possible after EndFrame() to improve parallelism between the GPU and the CPU.

Reimplemented from Base::RenderDeviceBase.

ImageFileFormat::Code Direct3D9::D3D9RenderDevice::SaveScreenshot(CoreGraphics::ImageFileFormat::Code fmt, const Ptr<IO::Stream> & out.)

save a screenshot to the provided stream

Save the backbuffer to the provided stream.

Reimplemented from Base::RenderDeviceBase.
void Base::RenderDeviceBase::SetOverrideDefaultRenderTarget(const Ptr<CoreGraphics::RenderTarget> &rt)

set an override for the default render target (call before Open())

Override the default render target (which is normally created in Open()) with a render target provided by the application, this is normally only useful for debugging and testing purposes.

void Base::RenderDeviceBase::AttachEventHandler(const Ptr<CoreGraphics::RenderEventHandler> &h) [inherited]

attach a render event handler

Attach an event handler to the render device.

void Base::RenderDeviceBase::RemoveEventHandler(const Ptr<CoreGraphics::RenderEventHandler> &h) [inherited]

remove a render event handler

Remove an event handler from the display device.

bool Base::RenderDeviceBase::NotifyEventHandlers(const CoreGraphics::RenderEvent &e) [protected, inherited]

notify event handlers about an event

Notify all event handlers about an event.

int Core::RefCounted::GetRefCount() const [inline, inherited]

getc the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**CoreGraphics::RenderEvent**
CoreGraphics::RenderEvent Class Reference

#include <renderevent.h>
Detailed Description

Render events are sent by the RenderDevice to registered render event handlers.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>event codes</em></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RenderEvent()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>RenderEvent(Code c)</code></td>
<td>constructor with event code</td>
</tr>
<tr>
<td><code>Code GetEventCode()</code> const</td>
<td>get event code</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
CoreGraphics::RenderEventHandler
CoreGraphics::RenderEventHandler
Class Reference

#include <rendereventhandler.h>

Inheritance diagram for CoreGraphics::RenderEventHandler:
Detailed Description

A render event handler object is notified by the `RenderDevice` about noteworthy events. To react to those events, derive a class from `RenderEventHandler`, and attach to the render device via `RenderDevice::AttachEventHandler()`.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RenderEventHandler ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~RenderEventHandler ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <code>OnAttach ()</code></td>
<td>Called when the event handler is attached to the <code>RenderDevice</code></td>
</tr>
<tr>
<td>virtual void <code>OnRemove ()</code></td>
<td>Called when the event handler is removed from the <code>RenderDevice</code></td>
</tr>
<tr>
<td>virtual bool <code>PutEvent (const RenderEvent &amp;event)</code></td>
<td>Called by <code>RenderDevice</code> when an event happens</td>
</tr>
<tr>
<td>int <code>GetRefCount ()</code> const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release ()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Rtti &amp;rtti)</code> const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::String &amp;className)</code> const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::FourCC &amp;classFourCC)</code> const</td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA (const Rtti &amp;rtti)</code> const</td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::String &amp;rttiName)</code> const</td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String &amp; GetClassName ()</code> const</td>
<td>Get the class name</td>
</tr>
</tbody>
</table>
Util::FourCC  **GetClassFourCC** () const

*get the class FourCC code*
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>virtual bool</th>
<th><strong>HandleEvent</strong> (const <strong>RenderEvent</strong> &amp;event)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>called when an event should be processed, override this method in your subclass</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

bool CoreGraphics::RenderEventHandler::PutEvent(const RenderEvent & e) [virtual]
called by RenderDevice when an event happens

This method is called by the RenderDevice when an event happens. The default behaviour of this class is to call the HandleEvent() method directly. Subclasses of RenderEventHandler may choose to implement a different behaviour.

Reimplemented in CoreGraphics::ThreadSafeRenderEventHandler.

bool CoreGraphics::RenderEventHandler::HandleEvent(const RenderEvent & e) [protected, virtual]
called when an event should be processed, override this method in your subclass

Handle a render event. This method is usually called by PutEvent(), but subclasses of RenderEventHandler may choose to implement a different behaviour. Override this method in your subclass to process the incoming event.

Reimplemented in CoreGraphics::ThreadSafeRenderEventHandler.

int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::RenderShape
CoreGraphics::RenderShape Class Reference

#include <rendershape.h>
Detailed Description

Describes a shape which is rendered through the ShapeRenderer singleton. Shape rendering may be relatively inefficient and should only be used for debug visualizations. Please note that vertex and index data will be copied into a memory stream, so that it is safe to release or alter the original data once the shape object has been created. You have to be aware of the performance and memory-footprint implications though. Shape objects can be copied, but they will share the internal vertex/index data copy.

(C) 2008 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>shape types</em></td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RenderShape ()</strong></td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><strong>RenderShape (Threading::ThreadId threadId, Type shapeType, const Math::matrix44 &amp;modelTransform, const Math::float4 &amp;color)</strong></td>
<td><em>shortcut constructor for simple shapes</em></td>
</tr>
<tr>
<td><strong>IsValid ()</strong></td>
<td><em>return true if object has been setup</em></td>
</tr>
<tr>
<td><strong>SetupSimpleShape (Threading::ThreadId threadId, Type shapeType, const Math::matrix44 &amp;modelTransform, const Math::float4 &amp;color)</strong></td>
<td><em>setup simple shape</em></td>
</tr>
<tr>
<td>*<em>SetupPrimitives (Threading::ThreadId threadId, const Math::matrix44 &amp;modelTransform, PrimitiveTopology::Code topology, SizeT numPrimitives, const void <em>vertices, SizeT vertexWidth, const Math::float4 &amp;color)</em></em></td>
<td><em>setup primitive batch (SLOW!)</em></td>
</tr>
<tr>
<td>**SetupIndexedPrimitives (Threading::ThreadId threadId, const Math::matrix44 &amp;modelTransform, PrimitiveTopology::Code topology, SizeT numPrimitives, const void <em>vertices, SizeT numVertices, SizeT vertexWidth, const void <em>indices, IndexType::Code indexType, const Math::float4 &amp;color)</em></em></td>
<td><em>setup indexed primitive batch (SLOW!)</em></td>
</tr>
<tr>
<td><strong>GetThreadId ()</strong></td>
<td><em>get the thread id of this shape</em></td>
</tr>
<tr>
<td><strong>GetShapeType ()</strong></td>
<td><em>get shape type</em></td>
</tr>
<tr>
<td><strong>GetModelTransform ()</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Threading::ThreadId**

**Type**

**const Math::matrix44 &**
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetTopology()</code> const</td>
<td>Get primitive topology</td>
</tr>
<tr>
<td><code>GetNumPrimitives()</code> const</td>
<td>Get number of primitives</td>
</tr>
<tr>
<td><code>GetVertexData()</code> const</td>
<td>Get pointer to vertex data (returns 0 if none exist)</td>
</tr>
<tr>
<td><code>GetVertexWidth()</code> const</td>
<td>Get vertex width in number of floats</td>
</tr>
<tr>
<td><code>GetNumVertices()</code> const</td>
<td>Get number of vertices (only for indexed primitives)</td>
</tr>
<tr>
<td><code>GetIndexData()</code> const</td>
<td>Get index data (returns 0 if none exists)</td>
</tr>
<tr>
<td><code>GetIndexType()</code> const</td>
<td>Get the index type (16 or 32 bit)</td>
</tr>
<tr>
<td><code>GetColor()</code> const</td>
<td>Get shape color</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
CoreGraphics::RenderTarget
CoreGraphics::RenderTarget Class Reference

#include <rendertarget.h>

Inheritance diagram for CoreGraphics::RenderTarget:
Detailed Description

A render targets wraps up to 4 color buffers and an optional depth/stencil buffer into a C++ object. The special default render target represents the backbuffer and default depth/stencil surface.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>ClearFlag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>clear flags</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void Setup()</code></td>
<td><code>setup the render target object</code></td>
</tr>
<tr>
<td><code>void Discard()</code></td>
<td><code>discard the render target object</code></td>
</tr>
<tr>
<td><code>void BeginPass()</code></td>
<td><code>begin a render pass</code></td>
</tr>
<tr>
<td><code>void EndPass()</code></td>
<td><code>end current render pass</code></td>
</tr>
<tr>
<td><code>void GenerateMipLevels()</code></td>
<td><code>generate mipmap levels</code></td>
</tr>
<tr>
<td><code>bool IsValid()</code> const</td>
<td><code>return true if valid (has been setup)</code></td>
</tr>
<tr>
<td><code>void SetDefaultRenderTarget(bool b)</code></td>
<td><code>set to true if default render target (only called by RenderDevice)</code></td>
</tr>
<tr>
<td><code>bool IsDefaultRenderTarget()</code> const</td>
<td><code>get default render target flag</code></td>
</tr>
<tr>
<td><code>void SetWidth(SizeT w)</code></td>
<td><code>set render target width</code></td>
</tr>
<tr>
<td><code>SizeT GetWidth()</code> const</td>
<td><code>get width of render target in pixels</code></td>
</tr>
<tr>
<td><code>void SetHeight(SizeT h)</code></td>
<td><code>set render target height</code></td>
</tr>
<tr>
<td><code>SizeT GetHeight()</code> const</td>
<td><code>get height of render target in pixels</code></td>
</tr>
<tr>
<td><code>void SetAntiAliasQuality(CoreGraphics::AntiAliasQuality::Code c)</code></td>
<td><code>set antialias quality</code></td>
</tr>
<tr>
<td><code>CoreGraphics::AntiAliasQuality::Code GetAntiAliasQuality()</code> const</td>
<td><code>get anti-alias-quality</code></td>
</tr>
<tr>
<td><code>void SetColorBufferFormat(CoreGraphics::PixelFormat::Code)</code></td>
<td><code>set color buffer format</code></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>colorFormat)</code></td>
<td>add a color buffer</td>
</tr>
<tr>
<td><code>CoreGraphics::PixelFormat::Code GetColorBufferFormat () const</code></td>
<td>get color buffer format at index</td>
</tr>
<tr>
<td><code>void AddDepthStencilBuffer ()</code></td>
<td>internally create a depth/stencil buffer</td>
</tr>
<tr>
<td><code>void AddSharedDepthStencilBuffer (const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp;rt)</code></td>
<td>use external depth-stencil buffer</td>
</tr>
<tr>
<td><code>bool HasDepthStencilBuffer () const</code></td>
<td>return true if the render target has a depth/stencil buffer</td>
</tr>
<tr>
<td><code>void SetMipMapsEnabled (bool b)</code></td>
<td>enable mipmap generation for this render target</td>
</tr>
<tr>
<td><code>bool AreMipMapsEnabled () const</code></td>
<td>get mipmap generation flag</td>
</tr>
<tr>
<td><code>void SetResolveTextureResourceId (const Resources::ResourceId &amp;resId)</code></td>
<td>set resolve texture resource id</td>
</tr>
<tr>
<td><code>const Resources::ResourceId &amp; GetResolveTextureResourceId () const</code></td>
<td>get resolve texture resource id</td>
</tr>
<tr>
<td><code>void SetResolveDepthTextureResourceId (const Resources::ResourceId &amp;resId)</code></td>
<td>set optional resolve depth texture resource id</td>
</tr>
<tr>
<td><code>const Resources::ResourceId &amp; GetResolveDepthTextureResourceId () const</code></td>
<td>get optional resolve depth texture resource id</td>
</tr>
<tr>
<td><code>void SetResolveTextureWidth (SizeT w)</code></td>
<td>set resolve texture width</td>
</tr>
<tr>
<td><code>SizeT GetResolveTextureWidth () const</code></td>
<td>get resolve texture width</td>
</tr>
<tr>
<td><code>void SetResolveTextureHeight (SizeT h)</code></td>
<td>set resolve texture height</td>
</tr>
<tr>
<td><code>SizeT GetResolveTextureHeight () const</code></td>
<td>get resolve texture height</td>
</tr>
<tr>
<td><code>void SetResolveTargetCpuAccess (bool b)</code></td>
<td>set resolve target CPU access</td>
</tr>
</tbody>
</table>
**set cpu access flag**

bool **GetResolveTargetCpuAccess** () const

get cpu access flag

void **SetMRTIndex** (IndexT i)

set optional MRT (Multiple Render Target) index, default is 0

IndexT **GetMRTIndex** () const

get multiple-render-target index

void **SetClearFlags** (uint clearFlags)

set clear flags

uint **GetClearFlags** () const

get clear flags

void **SetClearColor** (const Math::float4 &c)

set clear color

const Math::float4 & **GetClearColor** () const

get clear color

void **SetClearDepth** (float d)

set clear depth

float **GetClearDepth** () const

get clear depth

void **SetClearStencil** (uchar s)

set clear stencil value

uchar **GetClearStencil** () const

get clear stencil value

void **SetResolveRect** (const Math::rectangle< int > &r)

set the current resolve rectangle (in pixels)

const Math::rectangle< int > & **GetResolveRect** () const

get resolve rectangle

void **BeginBatch** (CoreGraphics::BatchType::Code batchType)

begin a batch

void **EndBatch** ()

end current batch

bool **HasResolveTexture** () const

return true if resolve texture is valid
const Ptr< CoreGraphics::Texture > & GetResolveTexture () const
get the resolve texture as Nebula texture object

bool HasCPUResolveTexture () const
return true if cpu access resolve texture is valid

const Ptr< CoreGraphics::Texture > & GetCPUResolveTexture () const
get the resolve texture as Nebula texture object

bool HasDepthResolveTexture () const
return true if resolve texture is valid

const Ptr< CoreGraphics::Texture > & GetDepthResolveTexture () const
get the resolve texture as Nebula texture object

virtual void ResolveDepthBuffer ()
resolve depth buffer

SizeT GetMemorySize () const
get byte size in memory, implemented in platform specific classes

void SetMemoryOffset (SizeT size)
set optional memory offset, not used by all platforms

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
**bool** **IsA** (const **Util::FourCC** &rttiFourCC) const

*return true if this object is instance of given class, or a derived class, by string*

**const** **Util::String** & **GetClassName** () const

*get the class name*

**Util::FourCC** **GetClassFourCC** () const

*get the class FourCC code*
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

void SetupMultiSampleType ()

setup compatible multisample type
Member Function Documentation

void Direct3D9::D3D9RenderTarget::SetupMultiSampleType() [protected, inherited]

setup compatible multisample type

Select the antialias parameters that most closely resemble the preferred settings in the DisplayDevice object.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
CoreGraphics::Shader
CoreGraphics::Shader Class Reference

#include <shader.h>

Inheritance diagram for CoreGraphics::Shader:
Detailed Description

A shader object manages the entire render state required to render a mesh.

(C) 2007 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource states (DO NOT CHANGE ORDER!)</td>
<td></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>Unload</strong> ()</td>
<td>Unload the resource, or cancel the load</td>
</tr>
<tr>
<td>ID3DXEffect * <strong>GetD3D9Effect</strong> () const</td>
<td>Get pointer to D3D effect</td>
</tr>
<tr>
<td><strong>CreateShaderInstance</strong></td>
<td>Create a shader instance from this shader</td>
</tr>
<tr>
<td>void <strong>DiscardShaderInstance</strong> Ptr&lt;* CoreGraphics::ShaderInstance&gt; &amp;inst</td>
<td>Discard a shader instance</td>
</tr>
<tr>
<td>const <strong>GetAllShaderInstances</strong></td>
<td>Get all instances</td>
</tr>
<tr>
<td>void <strong>SetAsyncEnabled</strong> (bool b)</td>
<td>Request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td>bool <strong>IsAsyncEnabled</strong> () const</td>
<td>Return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td>void <strong>Lock</strong> ()</td>
<td>Set locked to true</td>
</tr>
<tr>
<td>void <strong>Unlock</strong> ()</td>
<td>Set locked to false</td>
</tr>
<tr>
<td>bool <strong>IsLocked</strong> () const</td>
<td>Returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td>void <strong>SetResourceId</strong> (const ResourceId &amp;id)</td>
<td>Set the resource identifier</td>
</tr>
<tr>
<td>const ResourceId &amp; <strong>GetResourceId</strong> () const</td>
<td>Get the resource identifier</td>
</tr>
<tr>
<td>void <strong>SetLoader</strong> (const Ptr&lt;ResourceLoader&gt; &amp;loader)</td>
<td>Set optional resource loader</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetLoader()</code> const</td>
<td>get optional resource loader</td>
</tr>
<tr>
<td><code>SetSaver()</code> (const <code>Ptr&lt; ResourceSaver &gt;</code> &amp; saver)</td>
<td>set optional resource saver</td>
</tr>
<tr>
<td><code>GetSaver()</code> const</td>
<td>get optional resource saver</td>
</tr>
<tr>
<td><code>GetUseCount()</code> const</td>
<td>get current use count</td>
</tr>
<tr>
<td><code>Load()</code></td>
<td>load the resource</td>
</tr>
<tr>
<td><code>SetState(State s)</code></td>
<td>set current state (usually only called during <code>Load()</code>!)</td>
</tr>
<tr>
<td><code>GetState()</code> const</td>
<td>get current state</td>
</tr>
<tr>
<td><code>IsLoaded()</code> const</td>
<td>return true if current state is Loaded</td>
</tr>
<tr>
<td><code>IsPending()</code> const</td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td><code>LoadFailed()</code> const</td>
<td>return true if current state is Failed</td>
</tr>
<tr>
<td><code>Save()</code></td>
<td>save the resource</td>
</tr>
<tr>
<td><code>GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFO)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>IncrUseCount</strong></td>
<td>increment use count</td>
</tr>
<tr>
<td>void</td>
<td><strong>DecrUseCount</strong></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

**Resource::State**

`Resources::Resource::Load()` [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the `IsPending()` method will return true as long as the load is in progress, and `IsLoaded()` will become true when the loading process has finished. If the load has failed, `IsPending()` will switch to false and `IsLoaded()` will not be true.

```cpp
bool Resources::Resource::Save() [virtual, inherited]
```

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::ShaderFeature
CoreGraphics::ShaderFeature Class Reference

#include <shaderfeature.h>
Detailed Description

A set of shader features is used to identify a specific variation of a shader. For performance reasons, several shader features are combined into a bit mask, so that finding a matching variation can be done by bit-mask operations. Association of feature names to bit numbers must be done dynamically, since shader features are not hardcoded into Nebula, instead, applications can implement their own features without having to change the Nebula runtime.

In string form, a features mask consists of the shader feature names, separated by '|' characters:

"Depth|Skinning" "Alpha|Skinning|Unlit"

An application may define up to 32 unique features (corresponding to the number of bits in an unsigned int).

Note that all shader feature functionality is offered through ShaderServer methods (this has been done to automatically ensure thread safety for the shader feature code).

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>Type Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>typedef unsigned int</code> <strong>Mask</strong></td>
<td>A shader feature bit mask</td>
</tr>
<tr>
<td><code>typedef Util::StringAtom</code> <strong>Name</strong></td>
<td>A single shader feature name</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:42 2010
CoreGraphics::ShaderInstance
CoreGraphics::ShaderInstance Class Reference

#include <shaderinstance.h>

Inheritance diagram for CoreGraphics::ShaderInstance:
Detailed Description

A shader instance object is created from a shader and contains a local copy of the original shader state which can be modified through ShaderVariable objects. Shader instance objects are created directly through the shader server.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID3DXEffect * <strong>GetD3D9Effect</strong> () const</td>
<td>get pointer to d3d9 effect object</td>
</tr>
<tr>
<td>bool <strong>SelectActiveVariation</strong> (CoreGraphics::ShaderFeature::Mask featureMask)</td>
<td>select active variation by feature mask</td>
</tr>
<tr>
<td>SizeT <strong>Begin</strong> ()</td>
<td>begin rendering through the currently selected variation, returns no. passes</td>
</tr>
<tr>
<td>void <strong>BeginPass</strong> (IndexT passIndex)</td>
<td>begin pass</td>
</tr>
<tr>
<td>void <strong>Commit</strong> ()</td>
<td>commit changes before rendering</td>
</tr>
<tr>
<td>void <strong>EndPass</strong> ()</td>
<td>end pass</td>
</tr>
<tr>
<td>void <strong>End</strong> ()</td>
<td>end rendering through variation</td>
</tr>
<tr>
<td>void <strong>Discard</strong> ()</td>
<td>discard the shader instance, must be called when no longer needed</td>
</tr>
<tr>
<td>bool <strong>IsValid</strong> () const</td>
<td>return true if this object is valid</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::Shader &gt; &amp; <strong>GetOriginalShader</strong> () const</td>
<td>get pointer to original shader which created this instance</td>
</tr>
<tr>
<td>bool <strong>HasVariableByName</strong> (const CoreGraphics::ShaderVariable::Name &amp;n) const</td>
<td>return true if the shader instance has a variable by name</td>
</tr>
<tr>
<td>bool <strong>HasVariableBySemantic</strong> (const CoreGraphics::ShaderVariable::Semantic &amp;n) const</td>
<td>return true if shader has variable by semantic</td>
</tr>
<tr>
<td>SizeT <strong>GetNumVariables</strong> () const</td>
<td>get number of variables</td>
</tr>
</tbody>
</table>
const Ptr<CoreGraphics::ShaderVariable> & GetVariableByIndex (IndexT i) const
get a variable by index

const Ptr<CoreGraphics::ShaderVariable> & GetVariableByName (const CoreGraphics::ShaderVariable::Name &n) const
get a variable by name

const Ptr<CoreGraphics::ShaderVariable> & GetVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &s) const
get a variable by semantic

bool HasVariation (CoreGraphics::ShaderFeature::Mask featureMask) const
return true if variation exists by matching feature mask

SizeT GetNumVariations () const
get number of variations in the shader

const Ptr<CoreGraphics::ShaderVariation> & GetVariationByIndex (IndexT i) const
get shader variation by index

const Ptr<CoreGraphics::ShaderVariation> & GetVariationByFeatureMask (CoreGraphics::ShaderFeature::Mask featureMask) const
get shader variation by feature mask

const Ptr<CoreGraphics::ShaderVariation> & GetActiveVariation () const
get currently active variation

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::FourCC &classFourCC) const
| bool | IsA (const Rtti &rtti) const | return true if this object is instance of given class or a derived class |
| bool | IsA (const Util::String &rttiName) | return true if this object is instance of given class or a derived class, by string |
| bool | IsA (const Util::FourCC &rttiFourCC) | return true if this object is instance of given class or a derived class, by fourcc |
| const Util::String & | GetClassName () const | get the class name |
| Util::FourCC | GetClassFourCC () const | get the class FourCC code |
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>virtual void</th>
<th><strong>Setup</strong> (const Ptr&lt; CoreGraphics::Shader &gt; &amp;origShader)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>setup the shader instance from its original shader object</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Cleanup</strong> ()</td>
</tr>
<tr>
<td></td>
<td>cleanup the shader instance</td>
</tr>
<tr>
<td>void</td>
<td><strong>OnLostDevice</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called by d3d9 shader server when d3d9 device is lost</td>
</tr>
<tr>
<td>void</td>
<td><strong>OnResetDevice</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called by d3d9 shader server when d3d9 device is reset</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Direct3D9::D3D9ShaderInstance::Setup(const CoreGraphics::Shader origShader &)

setup the shader instance from its original shader object

This method is called by Shader::CreateInstance() to setup the new shader instance.

Reimplemented from Base::ShaderInstanceBase.

void Base::ShaderInstanceBase::Discard()

discard the shader instance, must be called when instance no longer needed

This method must be called when the object is no longer needed for proper cleanup.

int Core::RefCounted::GetRefCount() const

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef()

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release()

decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const **Util::String** &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

**Util::FourCC**
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::ShaderServer
CoreGraphics::ShaderServer Class Reference

#include <shaderserver.h>

Inheritance diagram for CoreGraphics::ShaderServer:

```
Core::RefCounted
  |
  +------------------
  |                  
Base::ShaderServerBase
  |                  
  +------------------
  |                  
Direct3D9::D3D9ShaderServer
  |                  
  +------------------
  |                  
CoreGraphics::ShaderServer
```
Detailed Description

The ShaderServer object loads the available shaders and can instantiate shaders for usage.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShaderServer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~ShaderServer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>bool Open ()</strong></td>
<td>open the shader server</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close the shader server</td>
</tr>
<tr>
<td>ID3DXEffectPool * <strong>GetD3D9EffectPool () const</strong></td>
<td>get pointer to global effect pool</td>
</tr>
<tr>
<td><strong>bool isOpen () const</strong></td>
<td>return true if the shader server is open</td>
</tr>
<tr>
<td><strong>bool HasShader (const Resources::ResourceId &amp;resId) const</strong></td>
<td>return true if a shader exists</td>
</tr>
<tr>
<td><strong>Ptr&lt; CoreGraphics::ShaderInstance &gt; CreateShaderInstance (const Resources::ResourceId &amp;resId)</strong></td>
<td>create a new shader instance</td>
</tr>
<tr>
<td><strong>const Util::Dictionary&lt; Resources::ResourceId, Ptr&lt; CoreGraphics::Shader &gt; &gt; &amp; GetAllShaders () const</strong></td>
<td>get all loaded shaders</td>
</tr>
<tr>
<td><strong>void SetActiveShaderInstance (const CoreGraphics::ShaderInstance &amp;shaderInst)</strong></td>
<td>set currently active shader instance</td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp; GetActiveShaderInstance () const</strong></td>
<td>get currently active shader instance</td>
</tr>
<tr>
<td><strong>void ResetFeatureBits ()</strong></td>
<td>reset the current feature bits</td>
</tr>
<tr>
<td><strong>void SetFeatureBits (CoreGraphics::ShaderFeature::Mask)</strong></td>
<td>set shader feature by bit mask</td>
</tr>
<tr>
<td><strong>void ClearFeatureBits (CoreGraphics::ShaderFeature::Mask)</strong></td>
<td>clear shader feature by bit mask</td>
</tr>
</tbody>
</table>
clear shader feature by bit mask

CoreGraphics::ShaderFeature::Mask GetFeatureBits () const
get the current feature mask

CoreGraphics::ShaderFeature::Mask FeatureStringToMask (const Util::String &str)
convert a shader feature string into a feature bit mask

Util::String FeatureMaskToString (CoreGraphics::ShaderFeature::Mask mask)
convert shader feature bit mask into string

void ApplyObjectId (IndexT i)
apply an object id

bool HasSharedVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &sem) const
return true if a shared variable exists by semantic

SizeT GetNumSharedVariables () const
get number of shared variables

const Ptr< CoreGraphics::ShaderVariable > & GetSharedVariableByIndex (IndexT i) const
get a shared variable by index

const Ptr< CoreGraphics::ShaderVariable > & GetSharedVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &sem) const
get a shared variable by semantic

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const &amp;classFourCC</code></td>
<td></td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class or derived class.</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of given class or derived class, by string.</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class or derived class, by fourcc.</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName ()</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC ()</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
create a new shader instance

This creates a clone of a template shader. This is the only method to create a new shader object. When the shader instance is no longer needed, call UnregisterShaderInstance() for proper cleanup.

get the current refcount

Return the current refcount of the object.

increment refcount by one

Increment the refcount of the object.

decrement refcount and destroy object if refcount is zero

Get the class name

Get the class name of the object.
get the class FourCC code

Get the class FourCC of the object.

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::ShaderVariable
CoreGraphics::ShaderVariable Class Reference

#include <shadervariable.h>

Inheritance diagram for CoreGraphics::ShaderVariable:
Detailed Description

Provides direct access to a shader's global variable. The fastest way to change the value of a shader variable is to obtain a pointer to a shader variable once, and use it repeatedly to set new values.

(C) 2007 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>shader variable types</td>
</tr>
<tr>
<td>typedef Util::StringAtom</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>shader variable name typedef</td>
</tr>
<tr>
<td>typedef Util::StringAtom</td>
<td>Semantic</td>
</tr>
<tr>
<td></td>
<td>shader variable semantic typedef</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>SetInt</code></td>
<td>int value</td>
<td>set int value</td>
</tr>
<tr>
<td>void</td>
<td><code>SetIntArray</code></td>
<td>const int *values, SizeT count</td>
<td>set int array values</td>
</tr>
<tr>
<td>void</td>
<td><code>SetFloat</code></td>
<td>float value</td>
<td>set float value</td>
</tr>
<tr>
<td>void</td>
<td><code>SetFloatArray</code></td>
<td>const float *values, SizeT count</td>
<td>set float array values</td>
</tr>
<tr>
<td>void</td>
<td><code>SetFloat4</code></td>
<td>const Math::float4 &amp;value</td>
<td>set vector value</td>
</tr>
<tr>
<td>void</td>
<td><code>SetFloat4Array</code></td>
<td>const Math::float4 *values, SizeT count</td>
<td>set vector array values</td>
</tr>
<tr>
<td>void</td>
<td><code>SetMatrix</code></td>
<td>const Math::matrix44 &amp;value</td>
<td>set matrix value</td>
</tr>
<tr>
<td>void</td>
<td><code>SetMatrixArray</code></td>
<td>const Math::matrix44 *values, SizeT count</td>
<td>set matrix array values</td>
</tr>
<tr>
<td>void</td>
<td><code>SetBool</code></td>
<td>bool value</td>
<td>set bool value</td>
</tr>
<tr>
<td>void</td>
<td><code>SetBoolArray</code></td>
<td>const bool *values, SizeT count</td>
<td>set bool array values</td>
</tr>
<tr>
<td>void</td>
<td><code>SetTexture</code></td>
<td>const CoreGraphics::Texture &amp;value</td>
<td>set texture value</td>
</tr>
<tr>
<td></td>
<td><code>Ptr&lt; CoreGraphics::ShaderVariableInstance &gt; CreateInstance</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>shader variable instance</td>
<td><code>GetType()</code> const</td>
<td>get the data type of the variable</td>
<td></td>
</tr>
<tr>
<td><code>const Name</code> &amp;</td>
<td><code>GetName()</code> const</td>
<td>get the name of the variable</td>
<td></td>
</tr>
<tr>
<td><code>const Semantic</code> &amp;</td>
<td><code>GetSemantic()</code> const</td>
<td>get the semantics of the variable</td>
<td></td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>GetRefCount()</code> const</td>
<td>get the current refcount</td>
<td></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
<td></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
<td></td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
<td></td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsInstanceOf(const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
<td></td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
<td></td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsA(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
<td></td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsA(const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TypeToString</strong> (Type t)</td>
<td>convert type to string</td>
</tr>
<tr>
<td><strong>StringToType</strong> (const Util::String &amp;str)</td>
<td>convert string to type</td>
</tr>
<tr>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetType</strong> (Type t)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set variable type</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetName</strong> (const Name &amp;n)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set variable name</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetSemantic</strong> (const Semantic &amp;s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set variable semantic</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```
increment refcount by one

```
void Core::RefCounted::Release ( ) [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

```
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::ShaderVariableInstance
CoreGraphics::ShaderVariableInstance
Class Reference

#include <shadervariableinstance.h>

Inheritance diagram for CoreGraphics::ShaderVariableInstance:
Detailed Description

A ShaderVariableInstance associates a value with a ShaderVariable and can apply that value at any time to the ShaderVariable. Setting the value on a ShaderVariableInstance will just store the value but not change the actual ShaderVariable. Only calling Apply() will set the stored value on the ShaderVariable. ShaderVariableInstance objects are used to manage per-instance state when rendering ModelNodeInstances.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Setup</code>       (const <code>Ptr&lt;CoreGraphics::ShaderVariable&gt; &amp;var)</code></td>
<td>setup the object from a shader variable</td>
</tr>
<tr>
<td><code>Prepare</code>     (CoreGraphics::ShaderVariable::type)</td>
<td>prepare the object for late-binding</td>
</tr>
<tr>
<td><code>Bind</code>         (const <code>Ptr&lt;CoreGraphics::ShaderVariable&gt; &amp;var)</code></td>
<td>late-bind the variable instance to a shader variable</td>
</tr>
<tr>
<td><code>GetShaderVariable</code> () const</td>
<td>get the associated shader variable</td>
</tr>
<tr>
<td><code>Apply</code>        ()</td>
<td>apply local value to shader variable</td>
</tr>
<tr>
<td><code>SetInt</code>       (int value)</td>
<td>set int value</td>
</tr>
<tr>
<td><code>SetFloat</code>     (float value)</td>
<td>set float value</td>
</tr>
<tr>
<td><code>SetFloat4</code>    (const <code>Math::float4</code> &amp;value)</td>
<td>set float4 value</td>
</tr>
<tr>
<td><code>SetMatrix</code>    (const <code>Math::matrix44</code> &amp;value)</td>
<td>set matrix44 value</td>
</tr>
<tr>
<td><code>SetBool</code>      (bool value)</td>
<td>set bool value</td>
</tr>
<tr>
<td><code>SetTexture</code>   (const <code>Ptr&lt;CoreGraphics::Texture&gt; &amp;value)</code></td>
<td>set texture value</td>
</tr>
<tr>
<td><code>SetValue</code>     (const <code>Util::Variant</code> &amp;v)</td>
<td>set value directly</td>
</tr>
<tr>
<td><code>GetRefCount</code>  () const</td>
<td>get the current refcount</td>
</tr>
</tbody>
</table>
void AddRef ()
 increment refcount by one

void Release ()
 decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
 return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
 return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
 return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
 return true if this object is instance of given class or a derived class

bool IsA (const Util::String &rttiName) const
 return true if this object is instance of given class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
 return true if this object is instance of given class, by fourcc

const Util::String & GetClassName () const
 get the class name

Util::FourCC GetClassFourCC () const
 get the class FourCC code
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Base::ShaderVariableInstanceBase::Apply() [inherited]

apply local value to shader variable

Todo:
: hmm, the dynamic type switch is sort of lame...

int
Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::ShaderVariation
CoreGraphics::ShaderVariation Class Reference

#include <shadervariation.h>

Inheritance diagram for CoreGraphics::ShaderVariation:
Detailed Description

A variation of a shader implements a specific feature set identified by a feature mask.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3DXHANDLE GetD3D9Technique() const</td>
<td>get the D3DX technique handle</td>
</tr>
<tr>
<td>ID3DXEffect * GetD3D9Effect() const</td>
<td>get the D3DX effect which owns this variation</td>
</tr>
<tr>
<td>const Name &amp; Get_Name() const</td>
<td>get the shader variation's name</td>
</tr>
<tr>
<td>CoreGraphics::ShaderFeature::Mask GetFeatureMask() const</td>
<td>get the feature bit mask of this variation</td>
</tr>
<tr>
<td>SizeT GetNumPasses() const</td>
<td>get number of passes in this variation</td>
</tr>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA(const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA(const Util::FourCC &amp;rttiFourCC)</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>const</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
</tr>
</tbody>
</table>

`return true if this object is instance of given class, or a derived class, by fourcc`

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class name</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>getClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void DumpRefCountingLeaks ()</code></td>
<td><code>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</code></td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>SetName</code> (const <code>Name</code> &amp;n)</td>
<td>set variation name</td>
</tr>
<tr>
<td>void</td>
<td><code>SetFeatureMask</code> (CoreGraphics::ShaderFeature::Mask m)</td>
<td>set feature bit mask of this variation</td>
</tr>
<tr>
<td>void</td>
<td><code>SetNumPasses</code> (SizeT n)</td>
<td>set number of passes</td>
</tr>
</tbody>
</table>
Member Function Documentation

```c++
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```c++
void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```c++
void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```c++
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

```c++
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```c++
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
```
builds only!

This method should be called as the very last before an application exits.
CoreGraphics::ShapeRenderer
CoreGraphics::ShapeRenderer Class Reference

#include <shaperenderer.h>

Inheritance diagram for CoreGraphics::ShapeRenderer:
Detailed Description

Render shapes for debug visualizations.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShapeRenderer ()</strong>&lt;br&gt;<em>constructor</em></td>
<td></td>
</tr>
<tr>
<td>virtual <strong>~ShapeRenderer ()</strong>&lt;br&gt;<em>destructor</em></td>
<td></td>
</tr>
<tr>
<td>void <strong>Open ()</strong>&lt;br&gt;<em>open the shape renderer</em></td>
<td></td>
</tr>
<tr>
<td>void <strong>Close ()</strong>&lt;br&gt;<em>close the shape renderer</em></td>
<td></td>
</tr>
<tr>
<td>void <strong>DrawShapes ()</strong>&lt;br&gt;<em>draw attached shapes and clear deferred stack, must be called inside render loop</em></td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsOpen ()</strong> const&lt;br&gt;<em>return true if open</em></td>
<td></td>
</tr>
<tr>
<td>void <strong>DeleteShapesByThreadId</strong> (Threading::ThreadId threadId)&lt;br&gt;<em>delete shapes of given thread id</em></td>
<td></td>
</tr>
<tr>
<td>void <strong>AddShape</strong> (const CoreGraphics::RenderShape &amp;shape)&lt;br&gt;<em>add a shape for deferred rendering (can be called from outside render loop)</em></td>
<td></td>
</tr>
<tr>
<td>void <strong>AddShapes</strong> (const Util::Array<a href="">CoreGraphics::RenderShape</a> &amp;shapeArray)&lt;br&gt;<em>add multiple shapes</em></td>
<td></td>
</tr>
<tr>
<td>void <strong>AddWireFrameBox</strong> (const Math::bbox &amp;boundingBox, const Math::float4 &amp;color, Threading::ThreadId threadId)&lt;br&gt;<em>add wireframe box</em></td>
<td></td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const&lt;br&gt;<em>get the current refcount</em></td>
<td></td>
</tr>
<tr>
<td>void <strong>AddRef</strong>&lt;br&gt;<em>increment refcount by one</em></td>
<td></td>
</tr>
<tr>
<td>void <strong>Release</strong>&lt;br&gt;<em>decrement refcount and destroy object if refcount is zero</em></td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

const Util::FourCC & Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
CoreGraphics::StreamMeshLoader
CoreGraphics::StreamMeshLoader
Class Reference

#include <streammeshloader.h>

Inheritance diagram for CoreGraphics::StreamMeshLoader:
Detailed Description

Resource loader to setup a Mesh object from a stream.

(C) 2008 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void SetUsage (Base::ResourceBase::Usage usage)</td>
<td>set the intended resource usage (default is UsageImmutable)</td>
</tr>
<tr>
<td>Base::ResourceBase::Usage GetUsage () const</td>
<td>get resource usage</td>
</tr>
<tr>
<td>void SetAccess (Base::ResourceBase::Access access)</td>
<td>set the intended resource access (default is AccessNone)</td>
</tr>
<tr>
<td>Base::ResourceBase::Access GetAccess () const</td>
<td>get the resource access</td>
</tr>
<tr>
<td>virtual bool CanLoadAsync () const</td>
<td>override in subclass: return true if asynchronous loading is supported (default is true)</td>
</tr>
<tr>
<td>virtual bool OnLoadRequested ()</td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td>virtual void OnLoadCancelled ()</td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td>virtual bool OnPending ()</td>
<td>call frequently while after OnLoadRequested() to put Resource into loaded state</td>
</tr>
<tr>
<td>virtual void OnAttachToResource (const Ptr&lt;Resource&gt; &amp;res)</td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>virtual void OnRemoveFromResource ()</td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>bool IsAttachedToResource () const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt;Resource&gt; &amp; GetResource () const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td>Resource::State GetState () const</td>
<td>return current state</td>
</tr>
<tr>
<td>virtual void Reset ()</td>
<td></td>
</tr>
</tbody>
</table>
resets loader-stats e.g. state

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

**Utilities**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks</th>
<th>()</th>
</tr>
</thead>
</table>

_dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)_
Protected Member Functions

```cpp
void SetState (Resource::State s)

set current state
```
**Member Function Documentation**

```cpp
bool Resources::StreamResourceLoader::CanLoadAsync() const [virtual, inherited]
```

override in subclass: return true if asynchronous loading is supported (default is true)

Indicate whether this resource loader supports asynchronous loading. The default is true. Override this method in a subclass and return false otherwise.

Reimplemented from `Resources::ResourceLoader`.
Reimplemented in `Direct3D9::D3D9StreamShaderLoader`.

```cpp
bool Resources::StreamResourceLoader::OnLoadRequested() [virtual, inherited]
```

called by resource when a load is requested

Handle the generic load request. In the asynchronous case, this method will fire a ReadStream message and go into pending state. The client will then call `OnPending()` periodically which checks whether the ReadStream message has been handled and continue accordingly. In the synchronous state the method will create an `IO::Stream` object and call the `SetupResourceFromStream()` method directly.

Reimplemented from `Resources::ResourceLoader`.
Reimplemented in `Resources::D3D9TextureStreamer`.

```cpp
void Resources::StreamResourceLoader::OnLoadCancelled() [virtual, inherited]
```

called by resource to cancel a pending load

This method is called when the currently pending asynchronous load
request should be cancelled.

Reimplemented from **Resources::ResourceLoader**.

```cpp
bool Resources::StreamResourceLoader::OnPending() [virtual, inherited]
```

call frequently while after **OnLoadRequested()** to put **Resource** into loaded state

This method is called periodically by the client when the resource is in pending state. The pending ReadStream message will be checked, and if it has been handled successfully the **SetupResourceFromStream()** method will be called and the **Resource** will go into Loaded state. If anything goes wrong, the resource will go into Failed state.

Reimplemented from **Resources::ResourceLoader**.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

\texttt{Util::FourCC}\nCore::RefCounted::GetClassFourCC\(\,\)\,\texttt{const [inline, inherited]}

get the class FourCC code

Get the class FourCC of the object.

\texttt{void}\nCore::RefCounted::DumpRefCountingLeaks\(\,\)\,[static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::StreamShaderLoader
CoreGraphics::StreamShaderLoader
Class Reference

#include <streamshaderloader.h>

Inheritance diagram for CoreGraphics::StreamShaderLoader:

- Core::RefCounted
- Resources::ResourceLoader
- Resources::StreamResourceLoader
- Direct3D9::D3D9StreamShaderLoader
- CoreGraphics::StreamShaderLoader
Detailed Description

Resource loader to setup a Shader object from a stream.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual bool CanLoadAsync () const</code></td>
<td>return true if asynchronous loading is supported</td>
</tr>
<tr>
<td><code>virtual bool OnLoadRequested ()</code></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td><code>virtual void OnLoadCancelled ()</code></td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td><code>virtual bool OnPending ()</code></td>
<td>call frequently while after <code>OnLoadRequested()</code> to put Resource into loaded state</td>
</tr>
<tr>
<td><code>virtual void OnAttachToResource (const Ptr&lt;Resource&gt; &amp;res)</code></td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td><code>virtual void OnRemoveFromResource ()</code></td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td><code>bool IsAttachedToResource () const</code></td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td><code>const Ptr&lt;Resource&gt; &amp; GetResource () const</code></td>
<td>get pointer to resource</td>
</tr>
<tr>
<td><code>Resource::State GetState () const</code></td>
<td>return current state</td>
</tr>
<tr>
<td><code>virtual void Reset ()</code></td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class by fourcc</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by string</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by fourcc</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

void SetState (Resource::State S)  

set current state
Member Function Documentation

bool Resources::StreamResourceLoader::OnLoadRequested() [virtual, inherited]
called by resource when a load is requested

Handle the generic load request. In the asynchronous case, this method will fire a ReadStream message and go into pending state. The client will then call OnPending() periodically which checks whether the ReadStream message has been handled and continue accordingly. In the synchronous state the method will create an IO::Stream object and call the SetupResourceFromStream() method directly.

Reimplemented from Resources::ResourceLoader.

Reimplemented in Resources::D3D9TextureStreamer.

void Resources::StreamResourceLoader::OnLoadCancelled() [virtual, inherited]
called by resource to cancel a pending load

This method is called when the currently pending asynchronous load request should be cancelled.

Reimplemented from Resources::ResourceLoader.

bool Resources::StreamResourceLoader::OnPending() [virtual, inherited]
call frequently while after OnLoadRequested() to put Resource into loaded state

This method is called periodically by the client when the resource is in pending state. The pending ReadStream message will be checked, and if it has been handled successfully the SetupResourceFromStream() method will be called and the
**Resource** will go into Loaded state. If anything goes wrong, the resource will go into Failed state.

Reimplemented from **Resources::ResourceLoader**.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::StreamTextureLoader
CoreGraphics::StreamTextureLoader Class Reference

#include <streamtextureloader.h>

Inheritance diagram for CoreGraphics::StreamTextureLoader:
Detailed Description

Resource loader for loading texture data from a Nebula3 stream. Supports synchronous and asynchronous loading.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool CanLoadAsync () const</td>
<td>Override in subclass: return true if asynchronous loading is supported (default is true)</td>
</tr>
<tr>
<td>virtual bool OnLoadRequested ()</td>
<td>Called by resource when a load is requested</td>
</tr>
<tr>
<td>virtual void OnLoadCancelled ()</td>
<td>Called by resource to cancel a pending load</td>
</tr>
<tr>
<td>virtual bool OnPending ()</td>
<td>Call frequently while after OnLoadRequested() to put Resource into loaded state</td>
</tr>
<tr>
<td>virtual void OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</td>
<td>Called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>virtual void OnRemoveFromResource ()</td>
<td>Called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>bool IsAttachedToResource () const</td>
<td>Return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetResource () const</td>
<td>Get pointer to resource</td>
</tr>
<tr>
<td>Resource::State GetState () const</td>
<td>Return current state</td>
</tr>
<tr>
<td>virtual void Reset ()</td>
<td>Resets loader-stats e.g. state</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

void SetState (Resource::State S)

set current state
# Member Function Documentation

```cpp
bool Resources::StreamResourceLoader::CanLoadAsync() const [virtual, inherited]
```

override in subclass: return true if asynchronous loading is supported (default is true)

Indicate whether this resource loader supports asynchronous loading. The default is true. Override this method in a subclass and return false otherwise.

Reimplemented from `Resources::ResourceLoader`.

Reimplemented in `Direct3D9::D3D9StreamShaderLoader`.

```cpp
bool Resources::StreamResourceLoader::OnLoadRequested() [virtual, inherited]
```

called by resource when a load is requested

Handle the generic load request. In the asynchronous case, this method will fire a ReadStream message and go into pending state. The client will then call `OnPending()` periodically which checks whether the ReadStream message has been handled and continue accordingly. In the synchronous state the method will create an `IO::Stream` object and call the `SetupResourceFromStream()` method directly.

Reimplemented from `Resources::ResourceLoader`.

Reimplemented in `Resources::D3D9TextureStreamer`.

```cpp
void Resources::StreamResourceLoader::OnLoadCancelled() [virtual, inherited]
```

called by resource to cancel a pending load

This method is called when the currently pending asynchronous load
request should be cancelled.

Reimplemented from Resources::ResourceLoader.

bool
Resources::StreamResourceLoader::OnPending() [virtual, inherited]

call frequently while after OnLoadRequested() to put Resource into loaded state

This method is called periodically by the client when the resource is in pending state. The pending ReadStream message will be checked, and if it has been handled successfully the SetupResourceFromStream() method will be called and the Resource will go into Loaded state. If anything goes wrong, the resource will go into Failed state.

Reimplemented from Resources::ResourceLoader.

int
Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
get the class name

Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC(const) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::StreamTextureSaver
CoreGraphics::StreamTextureSaver Class Reference

#include <streamtexturesaver.h>

Inheritance diagram for CoreGraphics::StreamTextureSaver:
Detailed Description

Allows to save texture data in a standard file format into a stream.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool</td>
<td>OnSave ()</td>
<td>called by resource when a save is requested</td>
</tr>
<tr>
<td>void</td>
<td>SetStream (const Ptr&lt; IO::Stream &gt;&amp;stream)</td>
<td>set stream to save to</td>
</tr>
<tr>
<td>const Ptr&lt; IO::Stream &gt; &amp;</td>
<td>GetStream () const</td>
<td>get save-stream</td>
</tr>
<tr>
<td>void</td>
<td>SetFormat (CoreGraphics::ImageFileFormat::Code fmt)</td>
<td>set file format (default is JPG)</td>
</tr>
<tr>
<td>CoreGraphics::ImageFileFormat::Code</td>
<td>GetFormat () const</td>
<td>get file format</td>
</tr>
<tr>
<td>void</td>
<td>SetMipLevel (IndexT mipLevel)</td>
<td>set the mip level to save (default is 0, the top level)</td>
</tr>
<tr>
<td>IndexT</td>
<td>GetMipLevel () const</td>
<td>get the mip level to save</td>
</tr>
<tr>
<td>virtual void</td>
<td>OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</td>
<td>called when the resource saver is attached to its resource</td>
</tr>
<tr>
<td>virtual void</td>
<td>OnRemoveFromResource ()</td>
<td>called when the resource saver is removed from its resource</td>
</tr>
<tr>
<td>bool</td>
<td>IsAttachedToResource () const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp;</td>
<td>GetResource () const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td>int</td>
<td>GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td>AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void</td>
<td>Release ()</td>
<td>decrement refcount and destroy object if refcount is</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
<th>get the class name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>DumpRefCountingLeaks()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
CoreGraphics::TextElement
CoreGraphics::TextElement Class Reference

#include <textelement.h>
Detailed Description

Describes a text element for the text renderer.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TextElement()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>TextElement(Threading::ThreadId threadId, const Util::String &amp;text, const Math::float4 &amp;color, const Math::float2 &amp;pos)</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>GetThreadId()</code></td>
<td>get thread id</td>
</tr>
<tr>
<td><code>getText()</code></td>
<td>get text</td>
</tr>
<tr>
<td><code>GetColor()</code></td>
<td>get color</td>
</tr>
<tr>
<td><code>GetPosition()</code></td>
<td>get position</td>
</tr>
</tbody>
</table>
CoreGraphics::TextRenderer
CoreGraphics::TextRenderer Class Reference

#include <textrenderer.h>

Inheritance diagram for CoreGraphics::TextRenderer:

```
  Core::RefCounted
   `-> Base::TextRendererBase
        `-> Direct3D9::D3D9TextRenderer
            `-> CoreGraphics::TextRenderer
```
Detailed Description

A simple text renderer for drawing text on screen. Only for debug purposes.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TextRenderer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~TextRenderer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Open ()</strong></td>
<td>open the device</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close the device</td>
</tr>
<tr>
<td><strong>void DrawTextElements ()</strong></td>
<td>draw the accumulated text</td>
</tr>
<tr>
<td><strong>bool IsOpen () const</strong></td>
<td>check if text renderer open</td>
</tr>
<tr>
<td><strong>void DeleteTextElementsByThreadId</strong></td>
<td>delete added text by thread id</td>
</tr>
<tr>
<td><strong>void AddTextElement (const CoreGraphics::TextElement &amp;textElement)</strong></td>
<td>add text element for rendering</td>
</tr>
<tr>
<td><strong>void AddTextElements (const Util::Array&lt;CoreGraphics::TextElement &gt; &amp;textElement)</strong></td>
<td>add multiple text elements for rendering</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Direct3D9::D3D9TextRenderer::DrawTextElements( ) [inherited]
draw the accumulated text
Draw buffered text. This method is called once per frame.
Reimplemented from Base::TextRendererBase.

int
Core::RefCounted::GetRefCount( ) const [inline, inherited]
get the current refcount
Return the current refcount of the object.

void
Core::RefCounted::AddRef( ) [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void
Core::RefCounted::Release( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]
get the class name
Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
CoreGraphics::Texture
CoreGraphics::Texture Class Reference

#include <texture.h>

Inheritance diagram for CoreGraphics::Texture:
Detailed Description

Front-end class for texture objects.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>Enum</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Type</strong></td>
<td>texture types</td>
</tr>
<tr>
<td>enum</td>
<td><strong>CubeFace</strong></td>
<td>cube map face</td>
</tr>
<tr>
<td>enum</td>
<td><strong>Usage</strong></td>
<td>resource usage flags</td>
</tr>
<tr>
<td>enum</td>
<td><strong>State</strong></td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unload</strong> ()</td>
<td>Unload the resource, or cancel the pending load.</td>
</tr>
<tr>
<td><strong>Map</strong> (IndexT mipLevel, MapType mapType, MapInfo &amp;outMapInfo)</td>
<td>Map a texture mip level for CPU access.</td>
</tr>
<tr>
<td><strong>Map</strong> (IndexT mipLevel, Access accessMode, MapInfo &amp;outMapInfo)</td>
<td>Map the a texture mip level for CPU access.</td>
</tr>
<tr>
<td><strong>Unmap</strong> (IndexT mipLevel)</td>
<td>Unmap texture after CPU access.</td>
</tr>
<tr>
<td><strong>MapCubeFace</strong> (CubeFace face, IndexT mipLevel, MapType mapType, MapInfo &amp;outMapInfo)</td>
<td>Map a cube map face for CPU access.</td>
</tr>
<tr>
<td><strong>MapCubeFace</strong> (CubeFace face, IndexT mipLevel, Access accessMode, MapInfo &amp;outMapInfo)</td>
<td>Map a cube map face for CPU access.</td>
</tr>
<tr>
<td><strong>UnmapCubeFace</strong> (CubeFace face, IndexT mipLevel)</td>
<td>Unmap cube map face after CPU access.</td>
</tr>
<tr>
<td>IDirect3D9BaseTexture9 * [GetD3D9BaseTexture] (IDirect3DBaseTexture9 *ptr)</td>
<td>Get d3d9 base texture pointer.</td>
</tr>
<tr>
<td>IDirect3DTexture9 * [GetD3D9Texture] (IDirect3DTexture9 *ptr)</td>
<td>Get d3d9 texture pointer.</td>
</tr>
<tr>
<td>IDirect3DCubeTexture9 * [GetD3D9CubeTexture] (IDirect3DCubeTexture9 *ptr)</td>
<td>Get d3d9 cube texture pointer.</td>
</tr>
<tr>
<td>void [SetupFromD3D9Texture] (IDirect3DTexture9 *ptr, const bool setLoaded=true)</td>
<td>Setup from a IDirect3DTexture9.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><code>void SetupFromD3D9CubeTexture(IDirect3DCubeTexture9 *ptr, const bool setLoaded=true)</code></td>
<td>setup from a <code>IDirect3DCubeTexture</code></td>
</tr>
<tr>
<td><code>void SetupFromD3D9VolumeTexture(IDirect3DVolumeTexture9 *ptr, const bool setLoaded=true)</code></td>
<td>setup from a <code>IDirect3DVolumeTexture</code></td>
</tr>
<tr>
<td><code>Type GetType() const</code></td>
<td>get texture type</td>
</tr>
<tr>
<td><code>SizeT GetWidth() const</code></td>
<td>get width of texture</td>
</tr>
<tr>
<td><code>SizeT GetHeight() const</code></td>
<td>get height of texture (if 2d or 3d texture)</td>
</tr>
<tr>
<td><code>SizeT GetDepth() const</code></td>
<td>get depth of texture (if 3d texture)</td>
</tr>
<tr>
<td><code>SizeT GetNumMipLevels() const</code></td>
<td>get number of mip levels</td>
</tr>
<tr>
<td><code>SizeT GetSkippedMips() const</code></td>
<td>get number of currently skipped mip levels</td>
</tr>
<tr>
<td><code>void SetSkippedMips(SizeT m)</code></td>
<td>set number of currently skipped mip levels</td>
</tr>
<tr>
<td><code>CoreGraphics::PixelFormat::Code GetPixelFormat() const</code></td>
<td>get pixel format of the texture</td>
</tr>
<tr>
<td><code>void SetUsage(Usage usage)</code></td>
<td>set resource usage type</td>
</tr>
<tr>
<td><code>Usage GetUsage() const</code></td>
<td>get resource usage type</td>
</tr>
<tr>
<td><code>void SetAccess(Access access)</code></td>
<td>set resource cpu access type</td>
</tr>
<tr>
<td><code>Access GetAccess() const</code></td>
<td>get cpu access type</td>
</tr>
<tr>
<td><code>void SetAsyncEnabled(bool b)</code></td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
</tbody>
</table>
| `bool IsAsyncEnabled() const` | return true if asynchronous resource loading is
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void Lock()</td>
<td>set locked to true</td>
</tr>
<tr>
<td>void Unlock()</td>
<td>set locked to false</td>
</tr>
<tr>
<td>bool IsLocked()</td>
<td>returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td>void SetResourceId(const ResourceId &amp;id)</td>
<td>set the resource identifier</td>
</tr>
<tr>
<td>const ResourceId &amp; GetResourceId() const</td>
<td>get the resource identifier</td>
</tr>
<tr>
<td>void SetLoader(const Ptr&lt;ResourceLoader&gt; &amp;loader)</td>
<td>set optional resource loader</td>
</tr>
<tr>
<td>const Ptr&lt;ResourceLoader&gt; &amp; GetLoader() const</td>
<td>get optional resource loader</td>
</tr>
<tr>
<td>void SetSaver(const Ptr&lt;ResourceSaver&gt; &amp;saver)</td>
<td>set optional resource saver</td>
</tr>
<tr>
<td>const Ptr&lt;ResourceSaver&gt; &amp; GetSaver() const</td>
<td>get optional resource saver</td>
</tr>
<tr>
<td>SizeT GetUseCount() const</td>
<td>get current use count</td>
</tr>
<tr>
<td>virtual State Load()</td>
<td>load the resource</td>
</tr>
<tr>
<td>void SetState(State s)</td>
<td>set current state (usually only called during Load())</td>
</tr>
<tr>
<td>State GetState() const</td>
<td>get current state</td>
</tr>
<tr>
<td>bool IsLoaded() const</td>
<td>return true if current state is Loaded</td>
</tr>
<tr>
<td>bool IsPending() const</td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td>bool LoadFailed() const</td>
<td>return true if current state is Failed</td>
</tr>
<tr>
<td>Function/Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual bool <strong>Save</strong> ()</td>
<td>save the resource</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetClassName</strong> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong> <strong>GetClassFourCC</strong> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void SetType (Type t)</td>
<td>set texture type</td>
</tr>
<tr>
<td>void SetWidth (SizeT w)</td>
<td>set texture width</td>
</tr>
<tr>
<td>void SetHeight (SizeT h)</td>
<td>set texture height</td>
</tr>
<tr>
<td>void SetDepth (SizeT d)</td>
<td>set texture depth</td>
</tr>
<tr>
<td>void SetNumMipLevels (SizeT n)</td>
<td>set number of mip levels</td>
</tr>
<tr>
<td>void SetPixelFormat (CoreGraphics::PixelFormat::Code f)</td>
<td>set pixel format</td>
</tr>
<tr>
<td>void IncrUseCount ()</td>
<td>increment use count</td>
</tr>
<tr>
<td>void DecrUseCount ()</td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
void Direct3D9::D3D9Texture::SetupFromD3D9Texture ( IDirect3DTexture9 *tex2D, const bool setLoaded = true ) [inherited]

setup from a IDirect3DTexture9

Helper method to setup the texture object from a D3D9 2D texture.

void Direct3D9::D3D9Texture::SetupFromD3D9CubeTexture ( IDirect3DCubeTexture9 *texCube, const bool setLoaded = true ) [inherited]

setup from a IDirect3DCubeTexture

Helper method to setup the texture object from a D3D9 cube texture.

void Direct3D9::D3D9Texture::SetupFromD3D9VolumeTexture ( IDirect3DVolumeTexture9 *texVolume, const bool setLoaded = true )

setup from a IDirect3DVolumeTexture

Helper method to setup the texture object from a D3D9 volume texture.

Resource::State Resources::Resource::Load ( ) [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded
asynchronously, the `IsPending()` method will return true as long as the load is in progress, and `IsLoaded()` will become true when the loading process has finished. If the load has failed, `IsPending()` will switch to false and `IsLoaded()` will not be true.

```cpp
bool Resources::Resource::Save() [virtual, inherited]
```

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.
Util::FourCC
core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

void
core::RefCounted::DumpRefCountingLeaks()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
CoreGraphics::ThreadSafeDisplayEventHandler
CoreGraphics::ThreadSafeDisplayEventHandler

Class Reference

#include <threadsafedisplayeventhandler.h>

Inheritance diagram for
CoreGraphics::ThreadSafeDisplayEventHandler:
Detailed Description

A thread-safe subclass of DisplayEventHandler. Allows to receive DisplayEvents from a different thread then the render thread. The producer thread calls the PutEvent() method to push new events into the event handler, these events will be stored in a

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ThreadSafeDisplayEventHandler ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~ThreadSafeDisplayEventHandler ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual bool <strong>PutEvent (const DisplayEvent &amp;event)</strong></td>
<td>Called by DisplayDevice when an event happens</td>
</tr>
<tr>
<td>void <strong>HandlePendingEvents ()</strong></td>
<td>Handle all pending events (called by consumer thread)</td>
</tr>
<tr>
<td>virtual void <strong>OnAttach ()</strong></td>
<td>Called when the event handler is attached to the DisplayDevice</td>
</tr>
<tr>
<td>virtual void <strong>OnRemove ()</strong></td>
<td>Called when the event handler is removed from the DisplayDevice</td>
</tr>
<tr>
<td>int <strong>GetRefCount () const</strong></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Util::String &amp;className)</strong></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA (const Rtti &amp;rtti) const</strong></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA (const Util::String &amp;rttiName) const</strong></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName() const</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC() const</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>virtual bool</th>
<th><strong>HandleEvent</strong> (const <strong>DisplayEvent</strong> &amp;event)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>called when an event should be processed, override this method in your subclass</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool CoreGraphics::ThreadSafeDisplayEventHandler::PutEvent (const DisplayEvent & e ) [virtual]
```
called by `DisplayDevice` when an event happens

Put an event into the event handler. This method is called by the render thread's `DisplayDevice`. Events are queued until the consumer thread processes them by calling `HandlePendingEvents()`.

Reimplemented from `CoreGraphics::DisplayEventHandler`.

```cpp
void CoreGraphics::ThreadSafeDisplayEventHandler::HandlePendingEvents ()
```
handle all pending events (called by consumer thread)

Process pending events. This method should be called frequently by the consumer thread. Pending events will be dequeued from the internal event queue and the `HandleEvent()` method will be called once per event.

```cpp
bool CoreGraphics::ThreadSafeDisplayEventHandler::HandleEvent (const DisplayEvent & e ) [protected, virtual]
```
called when an event should be processed, override this method in your subclass

Handle an event. This method is called in the consumer thread context from the `HandlePendingEvents()` method for each pending event. Override this method in your subclass to process the event.

Reimplemented from `CoreGraphics::DisplayEventHandler`.

Reimplemented in `Win32::Win32InputDisplayEventHandler`.

```cpp
int ( ) const [inline, inherited]
```
Core::RefCounted::GetRefCount

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::ThreadSafeRenderEventHandler
CoreGraphics::ThreadSafeRenderEventHandler

Class Reference

#include <threadsaferendereventhandler.h>

Inheritance diagram for
CoreGraphics::ThreadSafeRenderEventHandler:

```
Core::RefCounted

CoreGraphics::RenderEventHandler

CoreGraphics::ThreadSafeRenderEventHandler
```
Detailed Description

A thread-safe subclass of RenderEventHandler. Allows to receive RenderEvents from a different thread then the render thread. The producer thread calls the PutEvent() method to push new events into the event handler, these events will be stored in a

(C) 2007 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ThreadSafeRenderEventHandler()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~ThreadSafeRenderEventHandler()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual <code>bool PutEvent(const RenderEvent &amp;event)</code></td>
<td>Called by <code>RenderDevice</code> when an event happens</td>
</tr>
<tr>
<td>void <code>HandlePendingEvents()</code></td>
<td>Handle all pending events (called by consumer thread)</td>
</tr>
<tr>
<td>virtual void <code>OnAttach()</code></td>
<td>Called when the event handler is attached to the <code>RenderDevice</code></td>
</tr>
<tr>
<td>virtual void <code>OnRemove()</code></td>
<td>Called when the event handler is removed from the <code>RenderDevice</code></td>
</tr>
<tr>
<td>int <code>GetRefCount()</code> const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf(const Util::String &amp;className)</code> const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA(const Rtti &amp;rtti)</code> const</td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA(const Util::String &amp;rttiName)</code> const</td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA(const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>virtual bool</th>
<th><strong>HandleEvent</strong> (const <strong>RenderEvent</strong> &amp;event)</th>
</tr>
</thead>
</table>

*called when an event should be processed, override this method in your subclass*
Member Function Documentation

bool CoreGraphics::ThreadSafeRenderEventHandler::PutEvent (const RenderEvent & e ) [virtual]
called by RenderDevice when an event happens
Put an event into the event handler. This method is called by the render thread's RenderDevice. Events are queued until the consumer thread processes them by calling HandlePendingEvents().
Reimplemented from CoreGraphics::RenderEventHandler.

void CoreGraphics::ThreadSafeRenderEventHandler::HandlePendingEvents ( )
handle all pending events (called by consumer thread)
Process pending events. This method should be called frequently by the consumer thread. Pending events will be dequeued from the internal event queue and the HandleEvent() method will be called once per event.

bool CoreGraphics::ThreadSafeRenderEventHandler::HandleEvent (const RenderEvent & e ) [protected, virtual]
called when an event should be processed, override this method in your subclass
Handle an event. This method is called in the consumer thread context from the HandlePendingEvents() method for each pending event. Override this method in your subclass to process the event.
Reimplemented from CoreGraphics::RenderEventHandler.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef(); [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release(); [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName(); const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC(); const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks(); [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::TransformDevice
CoreGraphics::TransformDevice Class Reference

#include <transformdevice.h>

Inheritance diagram for CoreGraphics::TransformDevice:

```
CoreGraphics::TransformDevice
    `-- Core::RefCounted
        `-- Base::TransformDeviceBase
            `-- Win32::D3D9TransformDevice
```
**Detailed Description**

Manages global transform matrices and their combinations. **Input** transforms are the view transform (transforms from world to view space), the projection transform (describes the projection from view space into projection space (pre-div-z)) and the current model matrix (transforms from model to world space). From these input transforms, the **TransformDevice** computes all useful combinations and inverted versions.

(C) 2007 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TransformDevice()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~TransformDevice()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>open the transform device</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>close the transform device</td>
</tr>
<tr>
<td><code>ApplyViewSettings()</code></td>
<td>updates shared shader variables dependent on view matrix</td>
</tr>
<tr>
<td><code>ApplyModelTransforms</code></td>
<td>apply any model transform needed, implementation is platform dependend</td>
</tr>
<tr>
<td><code>IsOpen()</code> const</td>
<td>return true if device is open</td>
</tr>
<tr>
<td><code>SetProjTransform</code></td>
<td>set projection transform</td>
</tr>
<tr>
<td><code>GetProjTransform()</code></td>
<td>get current projection matrix</td>
</tr>
<tr>
<td><code>GetInvProjTransform()</code></td>
<td>get inverted projection transform</td>
</tr>
<tr>
<td><code>SetViewTransform</code></td>
<td>set view transform</td>
</tr>
<tr>
<td><code>GetViewTransform()</code></td>
<td>get view transform</td>
</tr>
<tr>
<td><code>GetInvViewTransform()</code></td>
<td>get current inverted view transform</td>
</tr>
<tr>
<td><code>GetViewProjTransform()</code></td>
<td>get current view-projection transform</td>
</tr>
<tr>
<td><code>SetModelTransform</code></td>
<td>set model transform</td>
</tr>
<tr>
<td><code>GetModelTransform()</code></td>
<td></td>
</tr>
</tbody>
</table>
get current model transform

const Math::matrix44 & GetInvModelTransform ()
get current inverted model transform

const Math::matrix44 & GetModelViewTransform ()
get current model-view matrix

const Math::matrix44 & GetInvModelViewTransform ()
get current inverted model-view-transform

const Math::matrix44 & GetModelViewProjTransform ()
get current model-view-projection transform

void SetFocalLength (const Math::float2 &len)
set focal length

const Math::float2 & GetFocalLength () const
get focal length

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name
<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
CoreGraphics::VertexBuffer
CoreGraphics::VertexBuffer Class Reference

#include <vertexbuffer.h>

Inheritance diagram for CoreGraphics::VertexBuffer:
Detailed Description

A resource which holds an array of vertices.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>resource usage flags</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>Unload</strong> ()</td>
<td>unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td>void * <strong>Map</strong> (MapType mapType)</td>
<td>map the vertices for CPU access</td>
</tr>
<tr>
<td>void <strong>Unmap</strong> ()</td>
<td>unmap the resource</td>
</tr>
<tr>
<td>void <strong>SetD3D9VertexBuffer</strong> (IDirect3DVertexBuffer9 *ptr)</td>
<td>set d3d9 vertex buffer pointer</td>
</tr>
<tr>
<td>IDirect3DVertexBuffer9 * <strong>GetD3D9VertexBuffer</strong> () const</td>
<td>get pointer to d3d9 vertex buffer object</td>
</tr>
<tr>
<td>void <strong>SetVertexLayout</strong> (const Ptr<a href="">CoreGraphics::VertexLayout</a> &amp;vertexLayout)</td>
<td>set vertex layout (set by resource loader)</td>
</tr>
<tr>
<td>const Ptr<a href="">CoreGraphics::VertexLayout</a> &amp; <strong>GetVertexLayout</strong> () const</td>
<td>get the vertex layout</td>
</tr>
<tr>
<td>void <strong>SetNumVertices</strong> (SizeT numVertices)</td>
<td>set number of vertices (set by resource loader)</td>
</tr>
<tr>
<td>SizeT <strong>GetNumVertices</strong> () const</td>
<td>get number of vertices in the buffer</td>
</tr>
<tr>
<td>void <strong>SetUsage</strong> (Usage usage)</td>
<td>set resource usage type</td>
</tr>
<tr>
<td>Usage <strong>GetUsage</strong> () const</td>
<td>get resource usage type</td>
</tr>
<tr>
<td>void <strong>SetAccess</strong> (Access access)</td>
<td>set resource cpu access type</td>
</tr>
<tr>
<td>Access <strong>GetAccess</strong> () const</td>
<td>get cpu access type</td>
</tr>
<tr>
<td>void <strong>SetAsyncEnabled</strong> (bool b)</td>
<td>request synchronous/asynchronous</td>
</tr>
<tr>
<td>Function / Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>bool IsAsyncEnabled()</code></td>
<td>const; return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>void Lock()</code></td>
<td>set locked to true</td>
</tr>
<tr>
<td><code>void Unlock()</code></td>
<td>set locked to false</td>
</tr>
<tr>
<td><code>bool IsLocked()</code></td>
<td>const; returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td><code>void SetResourceID(const ResourceID &amp;id)</code></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><code>const ResourceID &amp; GetResourceID()</code></td>
<td>const; get the resource identifier</td>
</tr>
<tr>
<td><code>void SetLoader(const Ptr&lt;ResourceLoader&gt; &amp;loader)</code></td>
<td>set optional resource loader</td>
</tr>
<tr>
<td><code>const Ptr&lt;ResourceLoader&gt; &amp; GetLoader()</code></td>
<td>const; get optional resource loader</td>
</tr>
<tr>
<td><code>void SetSaver(const Ptr&lt;ResourceSaver&gt; &amp;saver)</code></td>
<td>set optional resource saver</td>
</tr>
<tr>
<td><code>const Ptr&lt;ResourceSaver&gt; &amp; GetSaver()</code></td>
<td>const; get optional resource saver</td>
</tr>
<tr>
<td><code>SizeT GetUseCount()</code></td>
<td>const; get current use count</td>
</tr>
<tr>
<td><code>virtual State Load()</code></td>
<td>load the resource</td>
</tr>
<tr>
<td><code>void SetState(State s)</code></td>
<td>set current state (usually only called during <code>Load()</code>!)</td>
</tr>
<tr>
<td><code>State GetState()</code></td>
<td>const; get current state</td>
</tr>
<tr>
<td><code>bool IsLoaded()</code></td>
<td>const; return true if current state is <code>Loaded</code></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsPending() const</code></td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td><code>bool LoadFailed() const</code></td>
<td>return true if current state is Failed</td>
</tr>
<tr>
<td><code>virtual bool Save()</code></td>
<td>save the resource</td>
</tr>
<tr>
<td><code>int GetRefCount() const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName() const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td><em>get the class FourCC code</em></td>
<td></td>
</tr>
</tbody>
</table>
# Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void IncrUseCount ()</code></td>
<td>increment use count</td>
</tr>
<tr>
<td><code>void DecrUseCount ()</code></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Resource::State**

**Resources::Resource::Load** ( ) [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the **IsPending()** method will return true as long as the load is in progress, and **IsLoaded()** will become true when the loading process has finished. If the load has failed, **IsPending()** will switch to false and **IsLoaded()** will not be true.

**bool**

**Resources::Resource::Save** ( ) [virtual, inherited]

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

**int**

**Core::RefCounted::GetRefCount** ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

**void**

**Core::RefCounted::AddRef** ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

**void**

**Core::RefCounted::Release** ( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
CoreGraphics::VertexComponent
CoreGraphics::VertexComponent Class Reference

#include <vertexcomponent.h>
Detailed Description

Describes a single vertex component in a vertex layout description.

(C) 2006 Radon Labs GmbH
Public Member Functions

**VertexComponent ()**

default constructor

**VertexComponent (SemanticName semName_, IndexT semIndex_, Format format_, IndexT streamIndex_=0)**

constructor
CoreGraphics::VertexLayout
CoreGraphics::VertexLayout Class Reference

#include <vertexlayout.h>

Inheritance diagram for CoreGraphics::VertexLayout:

![Inheritance diagram]

- Core::RefCounted
- Base::VertexLayoutBase
- Win360::D3D9VertexLayout
- CoreGraphics::VertexLayout
Detailed Description

Describe the layout of vertices in a vertex buffer.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>Setup (const Util::Array<a href="">CoreGraphics::VertexComponent</a> &amp; setup)</td>
<td>setup the vertex layout</td>
</tr>
<tr>
<td>void</td>
<td>Discard ()</td>
<td>discard the vertex layout object</td>
</tr>
<tr>
<td>void</td>
<td>SetD3D9VertexDeclaration (IDirect3DVertexDeclaration9 * ptr)</td>
<td>set d3d9 vertex declaration pointer</td>
</tr>
<tr>
<td>bool</td>
<td>IsValid () const</td>
<td>return true if valid has been setup</td>
</tr>
<tr>
<td>SizeT</td>
<td>GetNumComponents () const</td>
<td>get number of components</td>
</tr>
<tr>
<td>const CoreGraphics::VertexComponent &amp; GetComponentAt (IndexT i)</td>
<td>get vertex component at index</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>HasComponent (CoreGraphics::VertexComponent::SemanticName semName, IndexT semIndex)</td>
<td>return true if vertex component exists</td>
</tr>
<tr>
<td>IndexT</td>
<td>FindComponent (CoreGraphics::VertexComponent::SemanticName semName, IndexT semIndex)</td>
<td>get index of vertex component by semantics</td>
</tr>
<tr>
<td>SizeT</td>
<td>GetVertexByteSize () const</td>
<td>get the vertex stride in number of bytes</td>
</tr>
<tr>
<td>const Util::Array<a href="">CoreGraphics::VertexComponent</a> &amp; GetVertexComponents ()</td>
<td>get vertex components</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td>AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void</td>
<td>Release ()</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>decrement refcount and destroy object if refcount is zero</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
<td></td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;rtti)</code></td>
<td>return true if this object is instance of given class by string</td>
<td></td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class or a derived class</td>
<td></td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rtti)</code></td>
<td>return true if this object is instance of given class or a derived class by string</td>
<td></td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rtti)</code></td>
<td>return true if this object is instance of given class or a derived class by fourcc</td>
<td></td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName ()</code></td>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC ()</code></td>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void <strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Static Protected Member Functions

<table>
<thead>
<tr>
<th>static Util::String</th>
<th><strong>BuildSignature</strong> (const Util::Array<a href="">CoreGraphics::VertexComponent</a> &amp;c)</th>
<th>get sharing signature for a set of vertex components</th>
</tr>
</thead>
</table>
Member Function Documentation

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
CoreGraphics::VertexLayoutServer
CoreGraphics::VertexLayoutServer
Class Reference

#include <vertexlayoutserver.h>

Inheritance diagram for CoreGraphics::VertexLayoutServer:

[Inheritance diagram with classes Core::RefCounted, Base::VertexLayoutServerBase, CoreGraphics::VertexLayoutServer]
Detailed Description

The **VertexLayoutServer** creates **VertexLayout** objects shared by their vertex component signature. On some platforms it is more efficient to share **VertexLayout** objects across meshes with identical vertex structure. Note that there is no way to manually discard vertex components. Vertex components stay alive for the life time of the application until the **Close()** method of the **VertexLayoutServer** is called.

(C) 2007 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Open()</code></td>
<td>Open the server</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>Close the server</td>
</tr>
<tr>
<td><code>IsOpen()</code> const</td>
<td>Return true if open</td>
</tr>
<tr>
<td><code>CreateSharedVertexLayout(const Util::Array&lt;CoreGraphics::VertexComponent&gt;&amp; vertexComponents)</code></td>
<td>Create shared vertex layout object</td>
</tr>
<tr>
<td><code>GetRefCount()</code> const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti&amp; rtti)</code> const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String&amp; className)</code> const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC&amp; classFourCC)</code> const</td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti&amp; rtti)</code> const</td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName)</code> const</td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Debug::ConsolePageHandler
Debug::ConsolePageHandler Class Reference

#include <consolepagehandler.h>

Inheritance diagram for Debug::ConsolePageHandler:
Detailed Description

Print console output to HTML page.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConsolePageHandler ()</strong></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td><strong>virtual ~ConsolePageHandler ()</strong></td>
<td><em>destructor</em></td>
</tr>
<tr>
<td><strong>virtual void HandleRequest (const Ptr&lt; Http::HttpRequest &gt;&amp; request)</strong></td>
<td>handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetName () const</strong></td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetDesc () const</strong></td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetRootLocation () const</strong></td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>bool IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>bool IsA (const Util::String &amp;rttiName) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp; GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HandlePendingRequests()</code></td>
<td>handle all pending requests, called by local-thread's <code>HttpServerProxy</code></td>
</tr>
<tr>
<td><code>PutRequest(const Ptr&lt;HttpRequest&gt; &amp;httpRequest)</code></td>
<td>put a request to the pending queue, called by <code>HttpServer</code> thread</td>
</tr>
<tr>
<td><code>SetName(const Util::String &amp;n)</code></td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><code>SetDesc(const Util::String &amp;d)</code></td>
<td>set human readable description</td>
</tr>
<tr>
<td><code>SetRootLocation(const Util::String &amp;l)</code></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
void
Http::HttpRequestHandler::HandlePendingRequests() [protected, inherited]

handle all pending requests, called by local-thread's **HttpServerProxy**

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

void
Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest> &httpRequest) [protected, inherited]

put a request to the pending queue, called by **HttpServer** thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

int
Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]

get the class name
Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**Debug::CorePageHandler**
Debug::CorePageHandler Class Reference

#include <stringatompagehandler.h>

Inheritance diagram for Debug::CorePageHandler:

- Core::RefCounted
- Debug::CorePageHandler
- Http::HttpRequestHandler
Detailed Description

Provide information about Core subsystem to debug http server.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CorePageHandler ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>HandleRequest (const Ptr&lt; Http::HttpRequest &gt; &amp;request)</strong></td>
<td>handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td><strong>GetName () const</strong></td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td><strong>GetDesc () const</strong></td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td><strong>GetRootLocation () const</strong></td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td><strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::String &amp;className)</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA (const Util::String &amp;rttiName) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void HandlePendingRequests()</code></td>
<td>handle all pending requests, called by local-thread's <code>HttpServerProxy</code></td>
</tr>
<tr>
<td><code>void PutRequest(const Ptr&lt; HttpRequest &gt; &amp;httpRequest)</code></td>
<td>put a request to the pending queue, called by <code>HttpServer</code> thread</td>
</tr>
<tr>
<td><code>void SetName(const Util::String &amp;n)</code></td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><code>void SetDesc(const Util::String &amp;d)</code></td>
<td>set human readable description</td>
</tr>
<tr>
<td><code>void SetRootLocation(const Util::String &amp;l)</code></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void
Http::HttpRequestHandler::HandlePendingRequests() [protected, inherited]
```
handle all pending requests, called by local-thread's `HttpServerProxy`

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

```cpp
void
Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest>& httpRequest) [protected, inherited]
```
put a request to the pending queue, called by `HttpServer` thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

```cpp
int
Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void
Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
get the class name

Get the class name of the object.

get the class FourCC code

Get the class FourCC of the object.

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Debug::DebugCounter
Debug::DebugCounter Class Reference

#include <debugcounter.h>

Inheritance diagram for Debug::DebugCounter:
Detailed Description

A debug counter for counting events.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DebugCounter ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~DebugCounter ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>Setup</strong> (const Util::StringAtom &amp;name)</td>
<td>setup the counter</td>
</tr>
<tr>
<td>void <strong>Discard</strong> ()</td>
<td>discard the counter</td>
</tr>
<tr>
<td>bool <strong>IsValid</strong> () const</td>
<td>return true if object has been setup</td>
</tr>
<tr>
<td>void <strong>Begin</strong> (bool reset=true)</td>
<td>begin counting, optionally reset the counter</td>
</tr>
<tr>
<td>void <strong>Reset</strong> ()</td>
<td>manually reset the counter to zero</td>
</tr>
<tr>
<td>void <strong>Incr</strong> (int amount)</td>
<td>increment the counter by a specific value</td>
</tr>
<tr>
<td>void <strong>Decr</strong> (int amount)</td>
<td>decrement the counter by a specific value</td>
</tr>
<tr>
<td>void <strong>Set</strong> (int val)</td>
<td>set the counter directly</td>
</tr>
<tr>
<td>void <strong>End</strong> ()</td>
<td>end counting, write current value to history</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; <strong>GetName</strong> () const</td>
<td>get the counter's name</td>
</tr>
<tr>
<td>int <strong>GetSample</strong> () const</td>
<td>get the most recent sample</td>
</tr>
<tr>
<td>Util::Array&lt; int &gt; <strong>GetHistory</strong> () const</td>
<td>get the counter's history</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Util::String</code> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Util::FourCC</code> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Util::String</code> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp; <code>GetClassName</code> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC</code> <code>GetClassFourCC</code> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong></th>
<th>()</th>
</tr>
</thead>
</table>

*dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)*
Member Function Documentation

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Debug::DebugFloat
#include <debugfloat.h>
Detailed Description

This class is supposed to make it easier, to restore the exact value of floating- point-based types, like vector, matrix44, float etc. for debugging. printf cuts values and rounds it. Also denormalized values might be printed as "0.0". Its particular useful, if a certain set of input- parameters lets a function crash. Just call printHex for floating-point- based params so they are being printed to stdout with its exact bit- pattern. That dump can be passed to restoreHex and you can debug that function with that particular input-set.

Printing a float4 to stdout works like this:

```cpp
float4 v(1.0f, 2.0f, 3.0f, 4.0f); DebugFloat::printHex(v, "v");
```

The output in stdout is this:

```plaintext
v: 0x3F800000, 0x40000000, 0x40400000, 0x40800000
```

To restore the values, just pass the output to restoreHex

```cpp
float4 v = DebugFloat::restoreHex(0x3F800000, 0x40000000, 0x40400000, 0x40800000);
```

Now v has the exact same value(bit pattern) as it had when being printed with printHex

There is also a normal "print" for each type. Quite useful is the quaternion-version, since it gives a more human-readable output than just the normal debugger.

(C) 2009 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>printHex</code></td>
<td>print float's bit pattern as hex to stdout</td>
</tr>
<tr>
<td><code>printHex</code></td>
<td>print float's bit pattern as hex to stdout</td>
</tr>
<tr>
<td><code>printHex</code></td>
<td>print float4's bit pattern as hex to stdout</td>
</tr>
<tr>
<td><code>printHex</code></td>
<td>print float4's bit pattern as hex to stdout</td>
</tr>
<tr>
<td><code>printHex</code></td>
<td>print quaternion's bit pattern as hex to stdout</td>
</tr>
<tr>
<td><code>printHex</code></td>
<td>print quaternion's bit pattern as hex to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print float's values plain to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print float's values plain to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print float4's values plain to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print float4's values plain to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print matrix's values plain to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print matrix's values plain to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print matrix's values plain to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print quaternion's values plain to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print quaternion's values plain to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print plane's values plain to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print plane's values plain to stdout</td>
</tr>
<tr>
<td><code>print</code></td>
<td>print plane's values plain to stdout</td>
</tr>
</tbody>
</table>
Debug::DebugGraphicsHandler
Debug::DebugGraphicsHandler Class Reference

#include <debuggraphicshandler.h>

Inheritance diagram for Debug::DebugGraphicsHandler:
Detailed Description

Handles debug-visualization messages in the graphics thread.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DebugGraphicsHandler()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~DebugGraphicsHandler()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual bool HandleMessage(const Ptr&lt;Messaging::Message&gt; &amp;msg)</code></td>
<td>Handle a message, return true if handled</td>
</tr>
<tr>
<td><code>virtual void DoWork()</code></td>
<td>Optional &quot;per-frame&quot; DoWork method for continuous handlers</td>
</tr>
<tr>
<td><code>void SetPerfHUDEnabled(bool b)</code></td>
<td>Enable/disable the debug perf-hud</td>
</tr>
<tr>
<td><code>void TogglePerfHUD()</code></td>
<td>Toggle the perf-hud</td>
</tr>
<tr>
<td><code>bool IsPerfHUDEnabled()</code></td>
<td>Return true if debug perf-hud is enabled</td>
</tr>
<tr>
<td><code>void SetCompanyName(const Util::StringAtom &amp;companyName)</code></td>
<td>Set the company name</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetCompanyName()</code></td>
<td>Get the company name</td>
</tr>
<tr>
<td><code>void SetAppName(const Util::StringAtom &amp;appName)</code></td>
<td>Set the application name</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetAppName()</code></td>
<td>Get the application name</td>
</tr>
<tr>
<td><code>virtual void Open()</code></td>
<td>Called once on startup</td>
</tr>
<tr>
<td><code>virtual void Close()</code></td>
<td>Called once before shutdown</td>
</tr>
<tr>
<td><code>bool isOpen()</code></td>
<td>Return true if open</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName() const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC() const</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

void Messaging::Handler::Open ( ) [virtual, inherited]
called once on startup

Open the handler. This method is called once after the handler has been attached to a port and before the first call to HandleMessage().

Reimplemented in Debug::DebugHandler, Http::HttpMessageHandler, IO::IoInterfaceHandler, Animation::AnimEventServer, and Graphics::GraphicsHandler.

void Messaging::Handler::Close ( ) [virtual, inherited]
called once before shutdown

Close the handler. This method is called once before the handler is detached from the port.

Reimplemented in Debug::DebugHandler, Http::HttpMessageHandler, IO::IoInterfaceHandler, Animation::AnimEventServer, and Graphics::GraphicsHandler.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.
void Core::RefCounted::Release

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Debug::DebugHandler
Debug::DebugHandler Class Reference

#include <debughandler.h>

Inheritance diagram for Debug::DebugHandler:
Detailed Description

The message handler for the debug interface. Just wakes up from time to time to check for incoming Http requests.

(C) 2008 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DebugHandler()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~DebugHandler()</td>
<td>destructor</td>
</tr>
<tr>
<td>Open()</td>
<td>open the handler</td>
</tr>
<tr>
<td>Close()</td>
<td>close the handler</td>
</tr>
<tr>
<td>DoWork()</td>
<td>do per-frame work</td>
</tr>
<tr>
<td>SetCompanyName(const Util::StringAtom &amp;companyName)</td>
<td>set the company name</td>
</tr>
<tr>
<td>GetCompanyName() const</td>
<td>get the company name</td>
</tr>
<tr>
<td>SetAppName(const Util::StringAtom &amp;appName)</td>
<td>set the application name</td>
</tr>
<tr>
<td>GetAppName() const</td>
<td>get the application name</td>
</tr>
<tr>
<td>IsOpen() const</td>
<td>return true if open</td>
</tr>
<tr>
<td>virtual bool HandleMessage(const Ptr&lt;Message&gt; &amp;msg)</td>
<td>handle a message, return true if handled</td>
</tr>
<tr>
<td>GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>IsInstanceOf(const Util::String &amp;className)</td>
<td></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>bool</th>
<th>const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class by string</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class by fourcc</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Debug::DebugHandler::Open() [virtual]

open the handler

This method runs in the Debug thread's context and sets up the required runtime.

Reimplemented from Messaging::Handler.

void Debug::DebugHandler::DoWork() [virtual]

do per-frame work

The per-frame method just checks periodically whether there are any pending HttpRequests to process...

Reimplemented from Interface::InterfaceHandlerBase.

bool Messaging::Handler::HandleMessage(const Ptr<Message>& msg) [virtual, inherited]

handle a message, return true if handled

Derive this method in a subclass to handle specific messages. The method should return true only if a message has been handled.

Reimplemented in Http::HttpMessageHandler, IO::IoInterfaceHandler, Animation::AnimEventServer, Debug::DebugGraphicsHandler, and Graphics::GraphicsHandler.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount
Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Debug::DebugInterface Class Reference

#include <debuginterface.h>

Inheritance diagram for Debug::DebugInterface:
Detailed Description

**Interface** object of the **Debug** subsystem. This just creates a **DebugHandler** which runs the **DebugServer** in its own thread.

(C) 2008 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DebugInterface ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~DebugInterface ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Open ()</code></td>
<td>Open the interface object</td>
</tr>
<tr>
<td><code>AttachHandler (const Ptr&lt;Messaging::Handler&gt; &amp;h)</code></td>
<td>Attach a handler to the port (call before open!)</td>
</tr>
<tr>
<td><code>GetCompanyName () const</code></td>
<td>Get the company name</td>
</tr>
<tr>
<td><code>GetAppName () const</code></td>
<td>Get the application name</td>
</tr>
<tr>
<td><code>GetRootDirectory () const</code></td>
<td>Get the root directory</td>
</tr>
<tr>
<td><code>SetHandlerThread (const Ptr&lt;HandlerThreadBase&gt; &amp;handlerThread)</code></td>
<td>Set pointer to handler thread object (must be derived from HandlerThreadBase)</td>
</tr>
<tr>
<td><code>GetHandlerThread () const</code></td>
<td>Get pointer to handler thread object</td>
</tr>
<tr>
<td><code>RemoveHandler (const Ptr&lt;Handler&gt; &amp;h)</code></td>
<td>Dynamically remove a handler from the port</td>
</tr>
<tr>
<td><code>Close ()</code></td>
<td>Close the async port</td>
</tr>
<tr>
<td><code>IsOpen () const</code></td>
<td>Return true if port is open</td>
</tr>
<tr>
<td><code>Send (const Ptr&lt;MESSAGETYPE&gt; &amp;msg)</code></td>
<td>Send an asynchronous message to the port</td>
</tr>
<tr>
<td><code>SendWait (const Ptr&lt;MESSAGETYPE&gt; &amp;msg)</code></td>
<td>Send an asynchronous message to the port</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>void <code>Wait (const Ptr&lt; MESSAGETYPE &gt; &amp;msg)</code></td>
<td>send a message and wait for completion</td>
</tr>
<tr>
<td>bool <code>Peek (const Ptr&lt; MESSAGETYPE &gt; &amp;msg)</code></td>
<td>wait for a message to be handled</td>
</tr>
<tr>
<td>void <code>Cancel (const Ptr&lt; MESSAGETYPE &gt; &amp;msg)</code></td>
<td>peek a message whether it has been handled</td>
</tr>
<tr>
<td>int <code>GetRefCount () const</code></td>
<td>cancel a pending message</td>
</tr>
<tr>
<td>void <code>AddRef ()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>Release ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC <code>GetClassFourCC () const</code></td>
<td></td>
</tr>
</tbody>
</table>
get the class FourCC code
Static Public Member Functions

static void **DumpRefCountingLeaks** ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

```cpp
void Messaging::AsyncPort::RemoveHandler(const Ptr<Handler> & h) [virtual, inherited]
dynamically remove a handler from the port
Dynamically remove a message handler.
```

```cpp
void Messaging::AsyncPort::Close() [virtual, inherited]
close the async port
Closes the async port.
Reimplemented in Graphics::GraphicsInterface.
```

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.
```
**get the class name**

Get the class name of the object.

**Util::String & Core::RefCounted::GetClassName**

**get the class FourCC code**

Get the class FourCC of the object.

**Util::FourCC Core::RefCounted::GetClassFourCC**

**dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)**

This method should be called as the very last before an application exits.

```cpp
const Util::String & Core::RefCounted::GetClassName

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Debug::DebugPageHandler
Debug::DebugPageHandler Class Reference

#include <debugpagehandler.h>

Inheritance diagram for Debug::DebugPageHandler:
Detailed Description

Http request handler for the Debug subsystem.

(C) 2008 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DebugPageHandler ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>HandleRequest (const Ptr&lt; Http::HttpRequest &gt; &amp;request)</strong></td>
<td>handle a http request</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetName ()</strong> const</td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetDesc ()</strong> const</td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetRootLocation ()</strong> const</td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td>int <strong>GetRefCount ()</strong> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Rtti &amp;rtti)</strong> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Util::String &amp;className)</strong> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA (const Rtti &amp;rtti)</strong> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA (const Util::String &amp;rttiName)</strong> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA (const Util::FourCC &amp;rttiFourCC)</strong> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>** GetName()** const</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>** GetClassFourCC()** const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComputeMinMaxAvgTimes</td>
<td>void ComputeMinMaxAvgTimes (const Util::Array<a href="">Timing::Time</a>&amp; times, Timing::Time&amp; outMin, Timing::Time&amp; outMax, Timing::Time&amp; outAvg) const compute the min, max and average time from an array of times</td>
</tr>
<tr>
<td>ComputeMinMaxAvgCounts</td>
<td>void ComputeMinMaxAvgCounts (const Util::Array&lt;int&gt;&amp; counterValues, int&amp; outMin, int&amp; outMax, int&amp; outAvg) const compute min, max and average values from an array of counter samples</td>
</tr>
<tr>
<td>HandlePendingRequests</td>
<td>void HandlePendingRequests () handle all pending requests, called by local-thread's HttpServerProxy</td>
</tr>
<tr>
<td>PutRequest</td>
<td>void PutRequest (const Ptr&lt;HttpRequest&gt;&amp; httpRequest) put a request to the pending queue, called by HttpServer thread</td>
</tr>
<tr>
<td>SetName</td>
<td>void SetName (const Util::String&amp; n) set human readable name of the request handler</td>
</tr>
<tr>
<td>SetDesc</td>
<td>void SetDesc (const Util::String&amp; d) set human readable description</td>
</tr>
<tr>
<td>SetRootLocation</td>
<td>void SetRootLocation (const Util::String&amp; l) set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Debug::DebugPageHandler::ComputeMinMaxAvgTimes(
    const Util::Array<Timing::Time> &times,
    Timing::Time &outMin,
    Timing::Time &outMax,
    Timing::Time &outAvg)
const [protected]

compute the min, max and average time from an array of times

Gets the min/max/avg time from an array of Time samples.

void Debug::DebugPageHandler::ComputeMinMaxAvgCounts(
    const Util::Array<int> &counterValues,
    int & outMin,
    int & outMax,
    int & outAvg)
const [protected]

compute min, max and average values from an array of counter samples

Gets the min/max/avg counter values from an array of counter samples.

void Http::HttpRequestHandler::HandlePendingRequests () [protected, inherited]

handle all pending requests, called by local-thread's HttpServerProxy

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.
void Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest> & httpRequest) [protected, inherited]

put a request to the pending queue, called by HttpServer thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Debug::DebugServer
Debug::DebugServer Class Reference

#include <debugserver.h>

Inheritance diagram for Debug::DebugServer:

```
Core::RefCounted

Debug::DebugServer
```

Detailed Description

The debug server singleton is visible from all threads and keeps track of debug timer and debug counters.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DebugServer</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~DebugServer</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td>bool <strong>IsOpen</strong> () const</td>
<td>return true if debug server is open</td>
</tr>
</tbody>
</table>
| void **RegisterDebugTimer** (const **Ptr<
  DebugTimer** > &timer)                     | register a debug timer                                                     |
| void **UnregisterDebugTimer** (const **Ptr<
  DebugTimer** > &timer)                    | unregister a debug timer                                                   |
| void **RegisterDebugCounter** (const **Ptr<
  DebugCounter** > &counter)                | register a debug counter                                                   |
| void **UnregisterDebugCounter** (const **Ptr<
  DebugCounter** > &counter)                | unregister a debug counter                                                 |
<p>| <strong>Util::Array</strong>&lt; <strong>Ptr&lt; DebugTimer</strong> &gt; &gt;&gt;    | <strong>GetDebugTimers</strong> () const                                                 | get all registered debug timers                                           |
| <strong>Util::Array</strong>&lt; <strong>Ptr&lt; DebugCounter</strong> &gt; &gt;&gt;   | <strong>GetDebugCounters</strong> () const                                              | get all registered debug counters                                         |
| <strong>Ptr&lt; DebugTimer</strong> &gt;                        | <strong>GetDebugTimerByName</strong> (const <strong>Util::StringAtom</strong> &amp;name) const           | get debug timer by name, returns invalid ptr if not exists                |
| <strong>Ptr&lt; DebugCounter</strong> &gt;                      | <strong>GetDebugCounterByName</strong> (const <strong>Util::StringAtom</strong> &amp;name) const         | get debug counter by name, returns invalid ptr if not exists             |
| int <strong>GetRefCount</strong> () const                 | get the current refcount                                                   |
| void <strong>AddRef</strong> ()                           | increment refcount by one                                                  |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName()</code> const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC()</code> const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

going the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Debug::DebugShapeRenderer
Debug::DebugShapeRenderer Class Reference

#include <debugshaperenderer.h>

Inheritance diagram for Debug::DebugShapeRenderer:
Detailed Description

Client-side proxy for rendering debug shapes. Packs shape render requests into a message which is sent to the render thread once per frame.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DebugShapeRenderer ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~DebugShapeRenderer ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>void DrawBox (const Math::matrix44 &amp;modelTransform, const Math::float4 &amp;color)</td>
<td>Draw a box</td>
</tr>
<tr>
<td>void DrawSphere (const Math::matrix44 &amp;modelTransform, const Math::float4 &amp;color)</td>
<td>Draw a sphere</td>
</tr>
<tr>
<td>void DrawCylinder (const Math::matrix44 &amp;modelTransform, const Math::float4 &amp;color)</td>
<td>Draw a cylinder</td>
</tr>
<tr>
<td>void DrawTorus (const Math::matrix44 &amp;modelTransform, const Math::float4 &amp;color)</td>
<td>Draw a torus</td>
</tr>
<tr>
<td>void DrawPrimitives (const Math::matrix44 &amp;modelTransform, CoreGraphics::PrimitiveTopology::Code topology, SizeT numPrimitives, const void *vertices, SizeT vertexWidth, const Math::float4 &amp;color)</td>
<td>Draw primitives</td>
</tr>
<tr>
<td>void DrawIndexedPrimitives (const Math::matrix44 &amp;modelTransform, CoreGraphics::PrimitiveTopology::Code topology, SizeT numPrimitives, const void *vertices, SizeT numVertices, SizeT vertexWidth, const void *indices, CoreGraphics::IndexType::Code indexType, const Math::float4 &amp;color)</td>
<td>Draw indexed primitives</td>
</tr>
<tr>
<td>void OnFrame ()</td>
<td>Call once per frame to send of accumulated draw commands</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td>AddRef ()</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td>increment refcount by one</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>Release ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::String &amp;className) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Debug::DebugShapeRenderer::DrawPrimitives ( const Math::matrix44 & mod
    CoreGraphics::PrimitiveTopology::Code topo
    SizeT num
    const void * verti
    SizeT vertx
    const Math::float4 & colo,
)

draw primitives

NOTE: this method copies the vertex data to a temporary buffer.

void Debug::DebugShapeRenderer::DrawIndexedPrimitives ( const Math::matrix44 &
    CoreGraphics::PrimitiveTopology::Code
top
    SizeT num
    const void * vertx
    SizeT vertx
    SizeT const void *
    CoreGraphics::IndexType::Code
    const Math::float4 &
)

draw indexed primitives

NOTE: this method copies the vertex and index data to a temporary buffer.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Debug::DebugTextRenderer
Debug::DebugTextRenderer Class Reference

#include <debugtextrenderer.h>

Inheritance diagram for Debug::DebugTextRenderer:
Detailed Description

Client-side proxy for rendering debug text. Packs text rendering requests into RenderDebugText messages and sends them off once per frame to the render thread.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DebugTextRenderer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~DebugTextRenderer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>void DrawText (const Util::String &amp;text, const Math::float4 &amp;color, const Math::float2 &amp;pos)</strong></td>
<td>draw text</td>
</tr>
<tr>
<td><strong>void DrawText3D (const Util::String &amp;text, const Math::float4 &amp;color, const Math::point &amp;pos)</strong></td>
<td>draw text</td>
</tr>
<tr>
<td><strong>void OnFrame ()</strong></td>
<td>call once per frame to send of accumulated draw commands</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>bool IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>bool IsA (const Util::String &amp;rttiName) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>bool IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String &amp;</code></td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td><code>Util::FourCC</code></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Debug::DebugTimer
#include <debugtimer.h>

Inheritance diagram for Debug::DebugTimer:

```
Core::RefCoutned

Debug::DebugTimer
```
Detailed Description

A debug timer for measuring time spent in code blocks.

(C) 2008 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DebugTimer ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~DebugTimer ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Setup (const Util::StringAtom &amp;timerName)</code></td>
<td>Setup the timer</td>
</tr>
<tr>
<td><code>Discard ()</code></td>
<td>Discard the timer</td>
</tr>
<tr>
<td><code>IsValid () const</code></td>
<td>Return true if this timer has been setup</td>
</tr>
<tr>
<td><code>Start ()</code></td>
<td>Start or continue the timer</td>
</tr>
<tr>
<td><code>Pause ()</code></td>
<td>Pause the timer</td>
</tr>
<tr>
<td><code>Stop ()</code></td>
<td>Stop the timer, writes sample to history</td>
</tr>
<tr>
<td><code>StartAccum ()</code></td>
<td>Start or continue the timer</td>
</tr>
<tr>
<td><code>StopAccum ()</code></td>
<td>Stop the timer, writes sample to history</td>
</tr>
<tr>
<td><code>ResetAccum ()</code></td>
<td>Stop the timer, writes sample to history</td>
</tr>
<tr>
<td><code>GetName () const</code></td>
<td>Get the timer name</td>
</tr>
<tr>
<td><code>GetSample () const</code></td>
<td>Get the most current sample</td>
</tr>
<tr>
<td><code>GetHistory () const</code></td>
<td>Get the timer's history</td>
</tr>
<tr>
<td><code>GetRefCount () const</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release ()</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Ptr&lt; DebugTimer &gt;</code></td>
<td><strong>CreateAsSingleton</strong> (const <code>Util::StringAtom</code> &amp;timerName)</td>
<td>create as singleton</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DestroySingleton</strong> (const <code>Util::StringAtom</code> &amp;timerName)</td>
<td>create as singleton</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Debug::DisplayPageHandler
Debug::DisplayPageHandler Class Reference

#include <displaypagehandler.h>

Inheritance diagram for Debug::DisplayPageHandler:
Detailed Description

Provide information about the display to the debug http server.

The DisplayPageHandler can also serve a screenshot:

http://host/display/screenshot?fmt=[format]

Where format is one of jpg,bmp,png.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DisplayPageHandler ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>handleRequest (const [Ptr][Http::HttpRequest] &amp;request)</strong></td>
<td>handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td><strong>GetName () const</strong></td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td><strong>GetDesc () const</strong></td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td><strong>GetRootLocation () const</strong></td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td><strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA (const Util::String &amp;rttiName) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
# Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HandlePendingRequests()</code></td>
<td>handle all pending requests, called by local-thread's <code>HttpServerProxy</code></td>
</tr>
<tr>
<td><code>PutRequest(const Ptr&lt;HttpRequest&gt; &amp;httpRequest)</code></td>
<td>put a request to the pending queue, called by <code>HttpServer</code> thread</td>
</tr>
<tr>
<td><code>SetName(const Util::String &amp;n)</code></td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><code>SetDesc(const Util::String &amp;d)</code></td>
<td>set human readable description</td>
</tr>
<tr>
<td><code>SetRootLocation(const Util::String &amp;l)</code></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Http::HttpRequestHandler::HandlePendingRequests () [protected, inherited]
handle all pending requests, called by local-thread's HttpServerProxy

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.
```

```cpp
void Http::HttpRequestHandler::PutRequest (const Ptr<HttpRequest> &httpRequest) [protected, inherited]
put a request to the pending queue, called by HttpServer thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.
```

```cpp
int Core::RefCounted::GetRefCount () const [inline, inherited]
get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef () [inline, inherited]
increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release () [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```
get the class name

Get the class name of the object.

get the class FourCC code

Get the class FourCC of the object.

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**Debug::GraphicsPageHandler**
Debug::GraphicsPageHandler Class Reference

#include <graphicspagehandler.h>

Inheritance diagram for Debug::GraphicsPageHandler:
Detailed Description

Provides debug information about (internal)graphics subsystem.

(C) 2008 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphicsPageHandler ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual void HandleRequest (const Ptr<a href="Http::HttpRequest">Http::HttpRequest</a> &amp;request)</td>
<td>handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td>const Util::String &amp; GetName () const</td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td>const Util::String &amp; GetDesc () const</td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td>const Util::String &amp; GetRootLocation () const</td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBUA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HandlePendingRequests()</code></td>
<td>handle all pending requests, called by local-thread's <code>HttpServerProxy</code></td>
</tr>
<tr>
<td><code>PutRequest(const Ptr&lt;HttpRequest&gt; &amp;httpRequest)</code></td>
<td>put a request to the pending queue, called by <code>HttpServer</code> thread</td>
</tr>
<tr>
<td><code>SetName(const Util::String &amp;n)</code></td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><code>SetDesc(const Util::String &amp;d)</code></td>
<td>set human readable description</td>
</tr>
<tr>
<td><code>SetRootLocation(const Util::String &amp;l)</code></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Http::HttpRequestHandler::HandlePendingRequests() [protected, inherited]
```

handle all pending requests, called by local-thread's `HttpServerProxy`

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

```cpp
void Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest> & httpRequest) [protected, inherited]
```

put a request to the pending queue, called by `HttpServer` thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Debug::HelloWorldRequestHandler
Debug::HelloWorldRequestHandler
Class Reference

#include <helloworldrequesthandler.h>

Inheritance diagram for Debug::HelloWorldRequestHandler:
Detailed Description

Most simple HttpRequestHandler possible. Invoke from web browser with

http://127.0.0.1:2100/helloworld

(C) 2008 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HelloWorldRequestHandler ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>handleRequest (const Ptr&lt; Http::HttpRequest &gt; &amp;request)</strong></td>
<td>handle a http request</td>
</tr>
<tr>
<td><strong>GetName () const</strong></td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td><strong>GetDesc () const</strong></td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td><strong>GetRootLocation () const</strong></td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td><strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti)</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::String &amp;className)</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA (const Rtti &amp;rtti)</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA (const Util::String &amp;rttiName)</strong></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA (const Util::FourCC &amp;rttiFourCC)</strong></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String &amp;</code></td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HandlePendingRequests()</code></td>
<td>handle all pending requests, called by local-thread's <code>HttpServerProxy</code></td>
</tr>
<tr>
<td><code>PutRequest(const Ptr&lt; HttpRequest &gt; &amp;httpRequest)</code></td>
<td>put a request to the pending queue, called by <code>HttpServer</code> thread</td>
</tr>
<tr>
<td><code>SetName(const Util::String &amp;n)</code></td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><code>SetDesc(const Util::String &amp;d)</code></td>
<td>set human readable description</td>
</tr>
<tr>
<td><code>SetRootLocation(const Util::String &amp;l)</code></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Http::HttpRequestHandler::HandlePendingRequests() [protected, inherited]
handle all pending requests, called by local-thread's HttpServerProxy
Handle all pending http requests in the pending queue. This method
must be called frequently from the thread which created this request
handler.

void Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest> &httpRequest) [protected,
inherited]
put a request to the pending queue, called by HttpServer thread
Put a http request into the request handlers message queue. This
method is meant to be called from another thread.

int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.
```
const `Util::String` &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name
Get the class name of the object.

`Util::FourCC`
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Debug::IoPageHandler
Debug::IoPageHandler Class Reference

#include <iopagehandler.h>

Inheritance diagram for Debug::IoPageHandler:

```
Core::RefCounted

Http::HttpRequestHandler

Debug::IoPageHandler
```
Detailed Description

Provide information about IO subsystem to debug http server.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IoPageHandler</strong></td>
<td>()</td>
</tr>
<tr>
<td>constructor</td>
<td></td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>HandleRequest</strong> (const Ptr<a href="Http::HttpRequest">Http::HttpRequest</a> &amp;request)</td>
</tr>
<tr>
<td></td>
<td>handle a http request, the handler is expected to fill the content stream</td>
</tr>
<tr>
<td></td>
<td>with response data</td>
</tr>
<tr>
<td>const <strong>string</strong></td>
<td><strong>GetName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td>const <strong>string</strong></td>
<td><strong>GetDesc</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td>const <strong>string</strong></td>
<td><strong>GetRootLocation</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get a resource location path which is accepted by the handler (e.g.</td>
</tr>
<tr>
<td></td>
<td>&quot;/display&quot;)</td>
</tr>
<tr>
<td>int</td>
<td><strong>GetRefCount</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td><strong>AddRef</strong> ()</td>
</tr>
<tr>
<td></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void</td>
<td><strong>Release</strong> ()</td>
</tr>
<tr>
<td></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class,</td>
</tr>
<tr>
<td></td>
<td>by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class,</td>
</tr>
<tr>
<td></td>
<td>by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBUA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

| void | HandlePendingRequests () | handle all pending requests, called by local-thread's HttpServerProxy |
| void | PutRequest (const Ptr< HttpRequest > &httpRequest) | put a request to the pending queue, called by HttpServer thread |
| void | SetName (const Util::String &n) | set human readable name of the request handler |
| void | SetDesc (const Util::String &d) | set human readable description |
| void | SetRootLocation (const Util::String &l) | set the root location of the request handler |
Member Function Documentation

void
Http::HttpRequestHandler::HandlePendingRequests( ) [protected, inherited]

handle all pending requests, called by local-thread's HttpServerProxy

Handle all pending http requests in the pending queue. This method
must be called frequently from the thread which created this request
handler.

void
Http::HttpRequestHandler::PutRequest( const Ptr< HttpRequest > httpRequest ) [protected,
inherited]

put a request to the pending queue, called by HttpServer thread

Put a http request into the request handlers message queue. This
method is meant to be called from another thread.

int
Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
Debug::MemoryPageHandler
Debug::MemoryPageHandler Class Reference

#include <memorypagehandler.h>

Inheritance diagram for Debug::MemoryPageHandler:
Detailed Description

Provide information about memory allocations to debug http server.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemoryPageHandler ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual void HandleRequest (const Ptr&lt; Http::HttpRequest &gt; &amp;request)</td>
<td>handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td>const Util::String &amp; GetName () const</td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td>const Util::String &amp; GetDesc () const</td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td>const Util::String &amp; GetRootLocation () const</td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td><code>get the class name</code></td>
</tr>
<tr>
<td><code>Util::FourCC</code></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><code>get the class FourCC code</code></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HandlePendingRequests ()</code></td>
<td>handle all pending requests, called by local-thread's <code>HttpServerProxy</code></td>
</tr>
<tr>
<td><code>PutRequest (const Ptr&lt; HttpRequest &gt; &amp;httpRequest)</code></td>
<td>put a request to the pending queue, called by <code>HttpServer</code> thread</td>
</tr>
<tr>
<td><code>SetName (const Util::String &amp;n)</code></td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><code>SetDesc (const Util::String &amp;d)</code></td>
<td>set human readable description</td>
</tr>
<tr>
<td><code>SetRootLocation (const Util::String &amp;l)</code></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

`void Http::HttpRequestHandler::HandlePendingRequests();` [protected, inherited]
handle all pending requests, called by local-thread's `HttpServerProxy`

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

`void Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest> &httpRequest);` [protected, inherited]
put a request to the pending queue, called by `HttpServer` thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

`int Core::RefCounted::GetRefCount();` const [inline, inherited]
get the current refcount

Return the current refcount of the object.

`void Core::RefCounted::AddRef();` [inline, inherited]
increment refcount by one

Increment the refcount of the object.

`void Core::RefCounted::Release();` [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const *Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

*Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Debug::MeshPageHandler
Debug::MeshPageHandler Class Reference

#include <meshpagehandler.h>

Inheritance diagram for Debug::MeshPageHandler:
Detailed Description

Provide an HTML debug page with information about shared mesh resources.

Usage: http://host/mesh - provide a list of all meshes with their properties http://host/mesh?meshinfo=[resId] - display information about specific mesh

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MeshPageHandler ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>HandleRequest (const Ptr&lt; Http::HttpRequest &gt; &amp;request)</strong></td>
<td>handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td><strong>GetName () const</strong></td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td><strong>GetDesc () const</strong></td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td><strong>GetRootLocation () const</strong></td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td><strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA (const Util::String &amp;rttiName) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void HandlePendingRequests()</td>
<td>handle all pending requests, called by local-thread's HttpServerProxy</td>
</tr>
<tr>
<td>void PutRequest (const Ptr&lt; HttpRequest &gt; &amp;httpRequest)</td>
<td>put a request to the pending queue, called by HttpServer thread</td>
</tr>
<tr>
<td>void SetName (const Util::String &amp;n)</td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td>void SetDesc (const Util::String &amp;d)</td>
<td>set human readable description</td>
</tr>
<tr>
<td>void SetRootLocation (const Util::String &amp;l)</td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
### Member Function Documentation

**void**

Http::HttpRequestHandler::HandlePendingRequests() [protected, inherited]

handle all pending requests, called by local-thread's **HttpServerProxy**

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

**void**

Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest> &httpRequest) [protected, inherited]

put a request to the pending queue, called by **HttpServer** thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

**int**

Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

**void**

Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

**void**

Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const `Util::String` &
Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

`Util::FourCC` Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Debug::MiniDump
Debug::MiniDump Class Reference

#include <minidump.h>

Inheritance diagram for Debug::MiniDump:

Win32::Win32MiniDump

Debug::MiniDump
Detailed Description

Support for generating mini dumps. Mini dumps are automatically created when n_assert() or n_error() triggers.

(C) 2007 Radon Labs GmbH

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Debug::ObjectInspectorHandler
Debug::ObjectInspectorHandler Class Reference

#include <objectinspectorhandler.h>

Inheritance diagram for Debug::ObjectInspectorHandler:
Detailed Description

Provide information about all registered game entities to the debug http server.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ObjectInspectorHandler ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>HandleRequest (const Ptr&lt; Http::HttpRequest &gt; &amp;request)</strong></td>
<td>handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td><strong>GetName () const</strong></td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td><strong>GetDesc () const</strong></td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td><strong>GetRootLocation () const</strong></td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td><strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::String &amp;className)</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA (const Util::String &amp;rttiName) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
<td></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HandlePendingRequests</td>
<td><code>handle all pending requests, called by local-thread's HttpServerProxy</code></td>
</tr>
<tr>
<td>PutRequest</td>
<td><code>put a request to the pending queue, called by HttpServer thread</code></td>
</tr>
<tr>
<td>SetName</td>
<td><code>set human readable name of the request handler</code></td>
</tr>
<tr>
<td>SetDesc</td>
<td><code>set human readable description</code></td>
</tr>
<tr>
<td>SetRootLocation</td>
<td><code>set the root location of the request handler</code></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Http::HttpRequestHandler::HandlePendingRequests() [protected, inherited]
```
handle all pending requests, called by local-thread's `HttpServerProxy`

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

```cpp
void Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest> &httpRequest) [protected, inherited]
```
put a request to the pending queue, called by `HttpServer` thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

going the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

going the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Debug::ShaderPageHandler
Debug::ShaderPageHandler Class Reference

#include <shaderpagehandler.h>

Inheritance diagram for Debug::ShaderPageHandler:

- Core::RefCounted
  - Http::HttpRequestHandler
  - Debug::ShaderPageHandler
Detailed Description

Provide a HTML debug page for shaders.

Usage: http://host/shader - list of all shaders http://host/shader?shaderinfo=[resId] - information about a specific shader

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShaderPageHandler ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual void HandleRequest (const Ptr&lt; Http::HttpRequest &gt; &amp;request)</td>
<td>handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td>const Util::String &amp; GetName () const</td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td>const Util::String &amp; GetDesc () const</td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td>const Util::String &amp; GetRootLocation () const</td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HandlePendingRequests</strong></td>
<td>handle all pending requests, called by local-thread's <em>HttpServerProxy</em></td>
</tr>
<tr>
<td><strong>PutRequest</strong></td>
<td>put a request to the pending queue, called by <em>HttpServer</em> thread</td>
</tr>
<tr>
<td><strong>SetName</strong></td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><strong>SetDesc</strong></td>
<td>set human readable description</td>
</tr>
<tr>
<td><strong>SetRootLocation</strong></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Http::HttpRequestHandler::HandlePendingRequests() [protected, inherited]
handle all pending requests, called by local-thread's **HttpServerProxy**
Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

void Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest> &httpRequest) [protected, inherited]
put a request to the pending queue, called by **HttpServer** thread
Put a http request into the request handlers message queue. This method is meant to be called from another thread.

int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.
const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Debug::StreamingTexturePageHandler
Debug::StreamingTexturePageHandler
Class Reference

#include <streamingtexturepagehandler.h>

Inheritance diagram for Debug::StreamingTexturePageHandler:
Detailed Description

Http page handler, that provides statistics and control of texture streaming.

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StreamingTexturePageHandler ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>HandleRequest (const Ptr&lt; Http::HttpRequest &gt; &amp;request)</strong></td>
<td>Handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td><strong>GetName () const</strong></td>
<td>Get a human readable name of the request handler</td>
</tr>
<tr>
<td><strong>GetDesc () const</strong></td>
<td>Get a human readable description of the request handler</td>
</tr>
<tr>
<td><strong>GetRootLocation () const</strong></td>
<td>Get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td><strong>GetRefCount () const</strong></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti)</strong></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::String &amp;className)</strong></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA (const Rtti &amp;rtti)</strong></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA (const Util::String &amp;rttiName)</strong></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA (const Util::FourCC &amp;rttiFourCC)</strong></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const</td>
<td><code>Util::String &amp;</code> GetClassName () const</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td></td>
<td><code>Util::FourCC</code> GetClassFourCC () const</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><code>DumpRefCountingLeaks()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <strong>HandlePendingRequests</strong> ()</td>
<td>handle all pending requests, called by local-thread's <code>HttpServerProxy</code></td>
</tr>
<tr>
<td>void <strong>PutRequest</strong> (const <code>Ptr&lt; HttpRequest &gt;</code> &amp;httpRequest)</td>
<td>put a request to the pending queue, called by <code>HttpServer</code> thread</td>
</tr>
<tr>
<td>void <strong>SetName</strong> (const <code>Util::String</code> &amp;n)</td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td>void <strong>SetDesc</strong> (const <code>Util::String</code> &amp;d)</td>
<td>set human readable description</td>
</tr>
<tr>
<td>void <strong>SetRootLocation</strong> (const <code>Util::String</code> &amp;l)</td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Http::HttpRequestHandler::HandlePendingRequests() [protected, inherited]
handle all pending requests, called by local-thread's HttpServerProxy
Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

```cpp
void Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest> & httpRequest) [protected, inherited]
put a request to the pending queue, called by HttpServer thread
Put a http request into the request handlers message queue. This method is meant to be called from another thread.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.
const `Util::String` &
Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

`Util::FourCC`
Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

`void`
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Debug::SvgTestPageHandler
Debug::SvgTestPageHandler Class Reference

#include <svgtestpagehandler.h>

Inheritance diagram for Debug::SvgTestPageHandler:
Detailed Description

A HTTP test page handler to test SVG rendering functionality.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong><code>SvgTestPageHandler()</code></strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong><code>virtual void HandleRequest(const Ptr&lt;Http::HttpRequest&gt; &amp;request)</code></strong></td>
<td>Handle a HTTP request, the handler is expected to fill the content stream with response data.</td>
</tr>
<tr>
<td><strong><code>const boost::String &amp; GetName()</code></strong></td>
<td>Get a human readable name of the request handler</td>
</tr>
<tr>
<td><strong><code>const boost::String &amp; GetDesc()</code></strong></td>
<td>Get a human readable description of the request handler</td>
</tr>
<tr>
<td><strong><code>const boost::String &amp; GetRootLocation()</code></strong></td>
<td>Get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td><strong><code>int GetRefCount()</code></strong></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><strong><code>void AddRef()</code></strong></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><strong><code>void Release()</code></strong></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong><code>bool IsInstanceOf(const Rtti &amp;rtti)</code></strong></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong><code>bool IsInstanceOf(const boost::String &amp;className)</code></strong></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong><code>bool IsInstanceOf(const boost::FourCC &amp;classFourCC)</code></strong></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong><code>bool IsA(const Rtti &amp;rtti)</code></strong></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong><code>bool IsA(const boost::String &amp;rttiName)</code></strong></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong><code>bool IsA(const boost::FourCC &amp;rttiFourCC)</code></strong></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
</table>

*dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)*
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HandlePendingRequests()</code></td>
<td>handle all pending requests, called by local-thread's <code>HttpServerProxy</code></td>
</tr>
<tr>
<td><code>PutRequest(constPtr&lt;HttpRequest&gt;&amp;httpRequest)</code></td>
<td>put a request to the pending queue, called by <code>HttpServer</code> thread</td>
</tr>
<tr>
<td><code>SetName(constUtil::String&amp;n)</code></td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><code>SetDesc(constUtil::String&amp;d)</code></td>
<td>set human readable description</td>
</tr>
<tr>
<td><code>SetRootLocation(constUtil::String&amp;l)</code></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Http::HttpRequestHandler::HandlePendingRequests() [protected, inherited]
```

handle all pending requests, called by local-thread's `HttpServerProxy`

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

```cpp
void Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest> &httpRequest) [protected, inherited]
```

put a request to the pending queue, called by `HttpServer` thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name
Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Debug::TexturePageHandler
Debug::TexturePageHandler Class Reference

#include <texturepagehandler.h>

Inheritance diagram for Debug::TexturePageHandler:
Detailed Description

Provide a HTML debug page with information about all shared texture resources.

Usage: **http://host/texture** - provide a list of all textures with their properties

**http://host/texture?img=[resId]&fmt=[fmt]** - retrieve an image of the texture

**http://host/texture?texinfo=[resId]** - build a HTML page about specific texture

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TexturePageHandler ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual void <strong>HandleRequest</strong> (const Ptr&lt; Http::HttpRequest &gt; &amp;request)</td>
<td>handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetName</strong> () const</td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetDesc</strong> () const</td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetRootLocation</strong> () const</td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void HandlePendingRequests()</code></td>
<td>handle all pending requests, called by local-thread's <code>HttpServerProxy</code></td>
</tr>
<tr>
<td><code>void PutRequest(const Ptr&lt; HttpRequest &gt;&amp; httpRequest)</code></td>
<td>put a request to the pending queue, called by <code>HttpServer</code> thread</td>
</tr>
<tr>
<td><code>void SetName(const Util::String &amp;n)</code></td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><code>void SetDesc(const Util::String &amp;d)</code></td>
<td>set human readable description</td>
</tr>
<tr>
<td><code>void SetRootLocation(const Util::String &amp;l)</code></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Http::HttpRequestHandler::HandlePendingRequests ( ) [protected, inherited]

handle all pending requests, called by local-thread's HttpServerProxy

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

void
Http::HttpRequestHandler::PutRequest
( const Ptr< HttpRequest > httpRequest ) [protected,
inherited]

put a request to the pending queue, called by HttpServer thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:42 2010
Debug::ThreadPageHandler
Debug::ThreadPageHandler Class Reference

#include <threadpagehandler.h>

Inheritance diagram for Debug::ThreadPageHandler:
Detailed Description

Displays info about currently running Nebula3 threads.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ThreadPageHandler ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>HandleRequest</strong> (const <strong>Ptr&lt;Http::HttpRequest &gt;</strong> &amp;request)</td>
<td>handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td><strong>GetName () const</strong></td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td><strong>GetDesc () const</strong></td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td><strong>GetRootLocation () const</strong></td>
<td>get a resource location path which is accepted by the handler (e.g. “/display”)</td>
</tr>
<tr>
<td><strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>
const Util::String & **GetClassName** () const

*get the class name*

Util::FourCC **GetClassFourCC** () const

*get the class FourCC code*
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><strong>HandlePendingRequests</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>handle all pending requests, called by local-thread's <strong>HttpServerProxy</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>PutRequest</strong> (const <strong>Ptr</strong>&lt; <strong>HttpRequest</strong> &gt; &amp;httpRequest)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>put a request to the pending queue, called by <strong>HttpServer</strong> thread</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetName</strong> (const <strong>Util::String</strong> &amp;n)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set human readable name of the request handler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetDesc</strong> (const <strong>Util::String</strong> &amp;d)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set human readable description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetRootLocation</strong> (const <strong>Util::String</strong> &amp;l)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

**void**

Http::HttpRequestHandler::HandlePendingRequests ( ) [protected, inherited]

handle all pending requests, called by local-thread's **HttpServerProxy**

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

**void**

Http::HttpRequestHandler::PutRequest ( const Ptr< HttpRequest > & httpRequest ) [protected, inherited]

put a request to the pending queue, called by **HttpServer** thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

**int**

Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

**void**

Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

**void**

Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const `Util::String` &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

`Util::FourCC`
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Direct3D9::D3D9DisplayDevice
Direct3D9::D3D9DisplayDevice Class Reference

#include <d3d9displaydevice.h>

Inheritance diagram for Direct3D9::D3D9DisplayDevice:
Detailed Description

Direct3D9 implementation of DisplayDevice class. Manages the application window.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9DisplayDevice ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~D3D9DisplayDevice ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>bool AdapterExists (CoreGraphics::Adapter::Code</strong></td>
<td>Check if the adapter actually exists</td>
</tr>
<tr>
<td><strong>Util::Array&lt; CoreGraphics::DisplayMode &gt;</strong></td>
<td>Get available display modes on given adapter</td>
</tr>
<tr>
<td><strong>bool SupportsDisplayMode (CoreGraphics::Adapter::Code</strong></td>
<td>Return true if a given display mode is supported</td>
</tr>
<tr>
<td><strong>CoreGraphics::DisplayMode</strong></td>
<td>Get current adapter display mode (i.e. the desktop display mode)</td>
</tr>
<tr>
<td><strong>CoreGraphics::AdapterInfo</strong></td>
<td>Get general info about display adapter</td>
</tr>
<tr>
<td><strong>virtual bool Open ()</strong></td>
<td>Open the display</td>
</tr>
<tr>
<td><strong>virtual void Close ()</strong></td>
<td>Close the display</td>
</tr>
<tr>
<td><strong>virtual void ProcessWindowMessages ()</strong></td>
<td>Process window system messages, call this method once per frame</td>
</tr>
</tbody>
</table>
**HWND**

```cpp
HWND GetHwnd () const
get the application window HWND
```

**void**

```cpp
SetAdapter (CoreGraphics::Adapter::Code a)
set display adapter (make sure adapter exists!)
```

**CoreGraphics::Adapter::Code**

```cpp
GetAdapter () const
get display adapter
```

**void**

```cpp
SetDisplayMode (const CoreGraphics::DisplayMode &m)
set display mode (make sure the display mode is supported!)
```

**const CoreGraphics::DisplayMode &**

```cpp
GetDisplayMode () const
get display mode
```

**void**

```cpp
SetAntiAliasQuality (CoreGraphics::AntiAliasQuality::Code aa)
set antialias quality
```

**CoreGraphics::AntiAliasQuality::Code**

```cpp
GetAntiAliasQuality () const
get antialias quality
```

**void**

```cpp
SetFullscreen (bool b)
set windowed/fullscreen mode
```

**bool**

```cpp
IsFullscreen () const
get windowed/fullscreen mode
```

**void**

```cpp
SetDisplayModeSwitchEnabled (bool b)
enable display mode switch when running fullscreen (default is true);
```

**bool**

```cpp
IsDisplayModeSwitchEnabled () const
is display mode switch enabled for fullscreen?
```

**void**

```cpp
SetTripleBufferingEnabled (bool b)
enable triple buffer for fullscreen (default is double buffering)
```

**bool**

```cpp
IsTripleBufferingEnabled () const
is triple buffer enabled for fullscreen?
```

**void**

```cpp
SetAlwaysOnTop (bool b)
set always-on-top behaviour
```

**bool**

```cpp
IsAlwaysOnTop () const
get always-on-top behaviour
```

**void**

```cpp
SetVerticalSyncEnabled (bool b)
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool <code>IsVerticalSyncEnabled()</code></td>
<td>get vertical sync flag</td>
</tr>
<tr>
<td>void <code>SetIconName(const Util::String &amp;s)</code></td>
<td>set optional window icon resource name</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetIconName()</code></td>
<td>get optional window icon resource name</td>
</tr>
<tr>
<td>void <code>SetParentWindow(void *h)</code></td>
<td>set optional parent window handle</td>
</tr>
<tr>
<td>void * <code>GetParentWindow()</code> const</td>
<td>get optional parent window handle</td>
</tr>
<tr>
<td>void <code>SetWindowTitle(const Util::String &amp;t)</code></td>
<td>set window title string (can be changed anytime)</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetWindowTitle()</code></td>
<td>get window title string</td>
</tr>
<tr>
<td>bool <code>IsOpen()</code> const</td>
<td>return true if display is currently open</td>
</tr>
<tr>
<td>void <code>AttachEventHandler(const Ptr&lt;CoreGraphics::DisplayEventHandler&gt; &amp;h)</code></td>
<td>attach a display event handler</td>
</tr>
<tr>
<td>void <code>RemoveEventHandler(const Ptr&lt;CoreGraphics::DisplayEventHandler&gt; &amp;h)</code></td>
<td>remove a display event handler</td>
</tr>
<tr>
<td>int <code>GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, derived class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
</table>

*dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)*
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual bool</td>
<td>OpenWindow ()</td>
<td><code>open the application window</code></td>
</tr>
<tr>
<td>Virtual void</td>
<td>CloseWindow ()</td>
<td><code>close the application window</code></td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnMinimized ()</td>
<td>called on <code>WM_SIZE</code> when window is minimized</td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnRestored ()</td>
<td>called on <code>WM_SIZE</code> when window is restored</td>
</tr>
<tr>
<td>Virtual bool</td>
<td>OnSetCursor ()</td>
<td>called on <code>WM_SETCURSOR</code></td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnPaint ()</td>
<td>called on <code>WM_PAINT</code></td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnSetFocus ()</td>
<td>called on <code>WM_SETFOCUS</code></td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnKillFocus ()</td>
<td>called on <code>WM_KILLFOCUS</code></td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnCloseRequested ()</td>
<td>called on <code>WM_CLOSE</code> to request if window should be closed</td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnToggleFullscreenWindowed ()</td>
<td>called when Alt-Enter is pressed</td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnKeyDown (WPARAM wParam)</td>
<td>called on <code>WM_KEYDOWN</code></td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnKeyUp (WPARAM wParam)</td>
<td>called on <code>WM_KEYUP</code></td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnChar (WPARAM wParam)</td>
<td>called on <code>WM_CHAR</code></td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnMouseButton (UINT uMsg, LPARAM lParam)</td>
<td>called on mouse button event</td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnMouseMove (LPARAM lParam)</td>
<td>called on <code>WM_MOUSEMOVE</code></td>
</tr>
<tr>
<td>Virtual void</td>
<td>OnMouseWheel (WPARAM wParam)</td>
<td>called on <code>WM_MOUSEWHEEL</code></td>
</tr>
<tr>
<td>Input::Key::Code</td>
<td>TranslateKeyCode (WPARAM wParam) const</td>
<td></td>
</tr>
</tbody>
</table>
translate a Windows virtual key code into a Nebula3 key code

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math::float2</td>
<td><strong>ComputeAbsMousePos</strong> (LPARAM lParam) const</td>
<td>compute absolute mouse position from lParam</td>
</tr>
<tr>
<td>Math::float2</td>
<td><strong>ComputeNormMousePos</strong> (const Math::float2 &amp;absMousePos) const</td>
<td>compute normalized mouse position from absolute mouse pos</td>
</tr>
<tr>
<td>bool</td>
<td><strong>NotifyEventHandlers</strong> (const CoreGraphics::DisplayEvent &amp;e)</td>
<td>notify event handlers about an event</td>
</tr>
</tbody>
</table>
### Static Protected Member Functions

<table>
<thead>
<tr>
<th>static LRESULT CALLBACK</th>
<th><strong>WinProc</strong> (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>the WinProc</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

bool
Direct3D9::D3D9DisplayDevice::AdapterExists
(CoreGraphics::Adapter::Code adapter)

check if the adapter actually exists

This method checks if the given adapter actually exists.
Reimplemented from Base::DisplayDeviceBase.

Util::Array< DisplayMode >
Direct3D9::D3D9DisplayDevice::GetAvailableDisplayModes
(CoreGraphics::Adapter::Code
  CoreGraphics::PixelFormat::Code
)

get available display modes on given adapter

Enumerate the available display modes on the given adapter in the
given pixel format. If the adapter doesn't exist on this machine, an
empty array is returned.
Reimplemented from Base::DisplayDeviceBase.

void
Win32::Win32DisplayDevice::ProcessWindowMessages
() [virtual, inherited]

process window system messages, call this method once per frame

Polls for and processes window messages. Call this message once
per frame in your render loop. If the user clicks the window close
button, or hits Alt-F4, a CloseRequested input event will be sent.
Reimplemented from Base::DisplayDeviceBase.

LRESULT CALLBACK
Win32::Win32DisplayDevice::WinProc
(HWND hWnd,
  UINT uMsg,
  WPARAM wParam,
  LPARAM lParam
) [static, protected,
the WinProc

The Nebula3 WinProc.

bool Win32::Win32DisplayDevice::OpenWindow() [protected, virtual, inherited]

open the application window

Open the application window.

void Win32::Win32DisplayDevice::CloseWindow() [protected, virtual, inherited]

close the application window

Close the application window.

Input::Key::Code Win32::Win32DisplayDevice::TranslateKeyCode(WPARAM wParam) const [protected, inherited]

translate a Windows virtual key code into a Nebula3 key code

Helper method which translates a Win32 virtual key code into a Nebula key code.

void Base::DisplayDeviceBase::SetWindowTitle(const Util::String &str) [inherited]

set window title string (can be changed anytime)

Set the window title. An application should be able to change the window title at any time, that's why this is a virtual method, so that a subclass may override it.

void Base::DisplayDeviceBase::AttachEventHandler(const CoreGraphics::DisplayEventHandler &h) [inherited]

attach a display event handler
Attach an event handler to the display device.

```cpp
void Base::DisplayDeviceBase::RemoveEventHandler(const Ptr<CoreGraphics::DisplayEventHandler>& h) [inherited]
```

remove a display event handler

Remove an event handler from the display device.

```cpp
bool Base::DisplayDeviceBase::NotifyEventHandlers(const CoreGraphics::DisplayEvent& e) [protected, inherited]
```

notify event handlers about an event

Notify all event handlers about an event.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String& Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name


Get the class name of the object.

`Util::FourCC`

Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

`void`

Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Direct3D9::D3D9ParticleRenderer
Direct3D9::D3D9ParticleRenderer Class Reference

#include <d3d9particlerenderer.h>

Inheritance diagram for Direct3D9::D3D9ParticleRenderer:
Detailed Description

Particle system renderer for D3D9/Xbox360. The renderer makes use of hardware instancing to prevent writing redundant data to dynamic vertex buffers.


(C) 2008 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3D9ParticleRenderer ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~D3D9ParticleRenderer ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void Setup ()</td>
<td>Setup the particle renderer</td>
</tr>
<tr>
<td>virtual void Discard ()</td>
<td>Discard the particle renderer</td>
</tr>
<tr>
<td>virtual void BeginAttach ()</td>
<td>Begin attaching visible particle systems</td>
</tr>
<tr>
<td>virtual void EndAttach ()</td>
<td>Finish attaching visible particle systems</td>
</tr>
<tr>
<td>IndexT GetCurParticleVertexIndex () const</td>
<td>Get the current vertex index</td>
</tr>
<tr>
<td>void AddCurParticleVertexIndex (IndexT add)</td>
<td>Add particle vertex index</td>
</tr>
<tr>
<td>void * GetCurVertexPtr ()</td>
<td>Get the current vertex pointer</td>
</tr>
<tr>
<td>void SetCurVertexPtr (void *ptr)</td>
<td>Set the current vertex pointer</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::VertexBuffer &gt; &amp;</td>
<td>GetParticleVertexBuffer () const</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::VertexBuffer &gt; &amp;</td>
<td>GetCornerVertexBuffer () const</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::IndexBuffer &gt; &amp;</td>
<td>GetCornerIndexBuffer () const</td>
</tr>
<tr>
<td>CoreGraphics::PrimitiveGroup &amp;</td>
<td>GetPrimitiveGroup ()</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::VertexLayout &gt; &amp;</td>
<td>GetVertexLayout ()</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>IsValid</td>
</tr>
</tbody>
</table>
return true if particle renderer has been setup

void AddVisibleParticleSystem (const Ptr<Particles::ParticleSystemInstance> &particleSystemInstance)
attach a visible particle system instance

bool IsInAttach () const
is renderer in attach?

void RenderParticleSystem (const Ptr<Particles::ParticleSystemInstance> &particleSystemInstance)
render particles of previously attached particle system

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Direct3D9::D3D9ParticleRenderer::BeginAttach() [virtual]

begin attaching visible particle systems

This method is called once per frame before visible particle systems are attached.

Reimplemented from Base::ParticleRendererBase.

void Direct3D9::D3D9ParticleRenderer::EndAttach() [virtual]

finish attaching visible particle systems

This method is called once per frame after visible particle systems are attached.

Reimplemented from Base::ParticleRendererBase.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Direct3D9::D3D9ParticleSystemInstance
Direct3D9::D3D9ParticleSystemInstance
Class Reference

#include <d3d9particlesysteminstance.h>
Detailed Description

The per-instance object of a ParticleSystem. This is where actual particles are created and updated.

(C) 2008 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th></th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D3D9ParticleSystemInstance ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual</td>
<td>~D3D9ParticleSystemInstance ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void</td>
<td>UpdateVertexStreams ()</td>
<td>generate vertex streams to render</td>
</tr>
<tr>
<td>virtual void</td>
<td>Render ()</td>
<td>render the particle system</td>
</tr>
</tbody>
</table>
Direct3D9::D3D9RenderDevice
Direct3D9::D3D9RenderDevice Class Reference

#include <d3d9renderdevice.h>

Inheritance diagram for Direct3D9::D3D9RenderDevice:
Detailed Description

Implements a RenderDevice on top of Direct3D9.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9RenderDevice ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~D3D9RenderDevice ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>IDirect3DDevice9 * <strong>GetDirect3DDevice ()</strong> const</td>
<td>get pointer to the d3d device</td>
</tr>
<tr>
<td><strong>bool Open ()</strong></td>
<td>open the device</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close the device</td>
</tr>
<tr>
<td><strong>bool BeginFrame ()</strong></td>
<td>begin complete frame</td>
</tr>
<tr>
<td><strong>void SetStreamSource (IndexT streamIndex, const Ptr<a href="">CoreGraphics::VertexBuffer</a> &amp;vb, IndexT offsetVertexIndex)</strong></td>
<td>set the current vertex stream source</td>
</tr>
<tr>
<td><strong>void SetVertexLayout (const Ptr<a href="">CoreGraphics::VertexLayout</a> &amp;vl)</strong></td>
<td>set current vertex layout</td>
</tr>
<tr>
<td><strong>void SetIndexBuffer (const Ptr<a href="">CoreGraphics::IndexBuffer</a> &amp;ib)</strong></td>
<td>set current index buffer</td>
</tr>
<tr>
<td><strong>void Draw ()</strong></td>
<td>draw current primitives</td>
</tr>
<tr>
<td><strong>void DrawIndexedInstanced (SizeT numInstances)</strong></td>
<td>draw indexed, instanced primitives (see method header for details)</td>
</tr>
<tr>
<td><strong>void EndPass ()</strong></td>
<td>end current pass</td>
</tr>
<tr>
<td><strong>void EndFrame ()</strong></td>
<td>end complete frame</td>
</tr>
<tr>
<td><strong>void Present ()</strong></td>
<td>present the rendered scene</td>
</tr>
</tbody>
</table>
**CoreGraphics::ImageFileFormat::Code**

SaveScreenshot (CoreGraphics::ImageFileFormat::fmt, const Ptr< IO::Stream > &outStream)

save a screenshot to the provided stream

**D3DPRESENT_PARAMETERS**

GetPresentParams () const

get the present parameters

void SetOverrideDefaultRenderTarget (Ptr< CoreGraphics::RenderTarget &rt)

set an override for the default render target (call before Open()!)

bool IsOpen () const

return true if currently open

void AttachEventHandler (const Ptr< CoreGraphics::RenderEventHandler &h>)

attach a render event handler

void RemoveEventHandler (const Ptr< CoreGraphics::RenderEventHandler &h>)

remove a render event handler

const Ptr< CoreGraphics::RenderTarget > & GetDefaultRenderTarget () const

get default render target

bool HasPassRenderTarget () const

is a pass rendertarget set?

const Ptr< CoreGraphics::RenderTarget > & GetPassRenderTarget () const

get current set pass render target

void BeginPass (const Ptr< CoreGraphics::RenderTarget > &rt, const Ptr< CoreGraphics::ShaderInstance > &passShader)

begin rendering a frame pass

void BeginPass (const Ptr< CoreGraphics::MultipleRenderTarget &mrt, const Ptr< CoreGraphics::ShaderInstance > &passShader)

begin rendering a frame pass with a multiple
### Render Target

**void** `BeginBatch` *(CoreGraphics::BatchType::Code batchType, const Ptr<CoreGraphics::ShaderInstance> &batchShader)*

*begin rendering a batch inside*

**const Ptr<CoreGraphics::VertexBuffer> &** `GetStreamVertexBuffer` *(IndexT streamIndex)* const

*get currently set vertex buffer*

**IndexT** `GetStreamVertexOffset` *(IndexT streamIndex)* const

*get currently set vertex stream offset*

**const Ptr<CoreGraphics::VertexLayout> &** `GetVertexLayout` *

*get current vertex layout*

**const Ptr<CoreGraphics::IndexBuffer> &** `GetIndexBuffer` *

*get current index buffer*

**void** `SetPrimitiveGroup` *(const CoreGraphics::PrimitiveGroup &pg)*

*set current primitive group*

**const CoreGraphics::PrimitiveGroup &** `GetPrimitiveGroup` *

*get current primitive group*

**void** `EndBatch` *

*end current batch*

**bool** `IsInBeginFrame` *

*check if inside BeginFrame*

**bool** `GetVisualizeMipMaps` *

*get visualization of mipmaps flag*

**void** `SetVisualizeMipMaps` *(bool val)*

*set visualization of mipmaps flag*

**int** `GetRefCount` *

*get the current refcount*

**void** `AddRef` *

*increment refcount by one*

**void** `Release` *

*decrement refcount and destroy object if refcount zero*
bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static bool</td>
<td><strong>CanCreate</strong> ()</td>
<td><em>test if a compatible render device can be created on this machine</em></td>
</tr>
<tr>
<td>static IDirect3D *</td>
<td><strong>GetDirect3D</strong> ()</td>
<td><em>get pointer to Direct3D interface, opens Direct3D if not happened yet</em></td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
**Static Public Attributes**

<table>
<thead>
<tr>
<th>static const IndexT</th>
<th><strong>MaxNumVertexStreams</strong> = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>max number of vertex streams</td>
<td></td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
bool NotifyEventHandlers (const CoreGraphics::RenderEvent &e)
notify event handlers about an event
```
Member Function Documentation

bool
Direct3D9::D3D9RenderDevice::CanCreate() [static]

test if a compatible render device can be created on this machine

Test if the right Direct3D version is installed by trying to open Direct3D.

Reimplemented from Base::RenderDeviceBase.

IDirect3D *
Direct3D9::D3D9RenderDevice::GetDirect3D() [static]

get pointer to Direct3D interface, opens Direct3D if not happened yet

Get a pointer to the Direct3D interface. Opens Direct3D if not happened yet.

IDirect3DDevice9 *
Direct3D9::D3D9RenderDevice::GetDirect3DDevice() const

get pointer to the d3d device

Return a pointer to d3d device. Asserts that the device exists.

bool
Direct3D9::D3D9RenderDevice::Open()

open the device

Open the render device. When successful, the RenderEvent::DeviceOpen will be sent to all registered event handlers after the Direct3D device has been opened.

Reimplemented from Base::RenderDeviceBase.

void
Direct3D9::D3D9RenderDevice::Close()
close the device

Close the render device. The RenderEvent::DeviceClose will be sent to all registered event handlers.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
bool Direct3D9::D3D9RenderDevice::BeginFrame()
```

begin complete frame

Begin a complete frame. Call this once per frame before any rendering happens. If rendering is not possible for some reason (e.g. a lost device) the method will return false. This method may also send the DeviceLost and DeviceRestored RenderEvents to attached event handlers.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
void Direct3D9::D3D9RenderDevice::SetStreamSource(IndexT streamIndex,
                                                   const Ptr<CoreGraphics::VertexBuffer> &vb,
                                                   IndexT offsetVertexInc)
```

set the current vertex stream source

Sets the vertex buffer to use for the next **Draw()**.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
void Direct3D9::D3D9RenderDevice::SetVertexLayout(const Ptr<CoreGraphics::VertexLayout> &vl)
```

set current vertex layout

Sets the vertex layout for the next **Draw()**
Reimplemented from **Base::RenderDeviceBase**.

```cpp
void Direct3D9::D3D9RenderDevice::SetIndexBuffer(const Ptr<CoreGraphics::IndexBuffer> &ib)
```

set current index buffer

Sets the index buffer to use for the next **Draw()**.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
void Direct3D9::D3D9RenderDevice::Draw()
```

draw current primitives

Draw the current primitive group. Requires a vertex buffer, an optional index buffer and a primitive group to be set through the respective methods. To use non-indexed rendering, set the number of indices in the primitive group to 0.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
void Direct3D9::D3D9RenderDevice::DrawIndexedInstanced(SizeT numInstances)
```

draw indexed, instanced primitives (see method header for details)

Draw N instances of the current primitive group. Requires the following setup:

- vertex stream 0: vertex buffer with instancing data, one vertex per instance
- vertex stream 1: vertex buffer with instance geometry data
- index buffer: index buffer for geometry data
- primitive group: the primitive group which describes one instance
- vertex declaration: describes a combined vertex from stream 0 and stream 1

Reimplemented from **Base::RenderDeviceBase**.
void Direct3D9::D3D9RenderDevice::EndPass()

direct current pass

End the current rendering pass. This will flush all texture stages in order to keep the d3d9 resource reference counts consistent without too much hassle.

Reimplemented from Base::RenderDeviceBase.

void Direct3D9::D3D9RenderDevice::EndFrame()

direct complete frame

End a complete frame. Call this once per frame after rendering has happened and before Present(), and only if BeginFrame() returns true.

Reimplemented from Base::RenderDeviceBase.

void Direct3D9::D3D9RenderDevice::Present()

present the rendered scene

NOTE: Present() should be called as late as possible after EndFrame() to improve parallelism between the GPU and the CPU.

Reimplemented from Base::RenderDeviceBase.

ImageFileFormat::Code Direct3D9::D3D9RenderDevice::SaveScreenshot(CoreGraphics::ImageFileFormat::Code fmt, const Ptr<IO::Stream> & out)

save a screenshot to the provided stream

Save the backbuffer to the provided stream.

Reimplemented from Base::RenderDeviceBase.
void Base::RenderDeviceBase::SetOverrideDefaultRenderTarget (const Ptr<CoreGraphics::RenderTarget> rt) &

set an override for the default render target (call before Open())!

Override the default render target (which is normally created in Open()) with a render target provided by the application, this is normally only useful for debugging and testing purposes.

void Base::RenderDeviceBase::AttachEventHandler (const Ptr<CoreGraphics::RenderEventHandler> h) [inherited]

attach a render event handler

Attach an event handler to the render device.

void Base::RenderDeviceBase::RemoveEventHandler (const Ptr<CoreGraphics::RenderEventHandler> h) [inherited]

remove a render event handler

Remove an event handler from the display device.

bool Base::RenderDeviceBase::NotifyEventHandlers (const CoreGraphics::RenderEvent e) [protected, inherited]

notify event handlers about an event

Notify all event handlers about an event.

int Core::RefCounted::GetRefCount () const [inline, inherited]

get the current refcount

Return the current refcount of the object.
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Direct3D9::D3D9RenderTarget
Direct3D9::D3D9RenderTarget Class Reference

#include <d3d9rendertarget.h>

Inheritance diagram for Direct3D9::D3D9RenderTarget:
Detailed Description

D3D9 implementation of RenderTarget.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>ClearFlag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>clear flags</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>D3D9RenderTarget ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~D3D9RenderTarget ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Setup ()</strong></td>
<td>setup the render target object</td>
</tr>
<tr>
<td><strong>Discard ()</strong></td>
<td>discard the render target object</td>
</tr>
<tr>
<td><strong>BeginPass ()</strong></td>
<td>begin a render pass</td>
</tr>
<tr>
<td><strong>EndPass ()</strong></td>
<td>end current render pass</td>
</tr>
<tr>
<td><strong>GenerateMipLevels ()</strong></td>
<td>generate mipmap levels</td>
</tr>
<tr>
<td><strong>IsValid () const</strong></td>
<td>return true if valid (has been setup)</td>
</tr>
<tr>
<td><strong>SetDefaultRenderTarget (bool b)</strong></td>
<td>set to true if default render target (only called by RenderDevice)</td>
</tr>
<tr>
<td><strong>IsDefaultRenderTarget () const</strong></td>
<td>get default render target flag</td>
</tr>
<tr>
<td><strong>setWidth (SizeT w)</strong></td>
<td>set render target width</td>
</tr>
<tr>
<td><strong>GetWidth () const</strong></td>
<td>get width of render target in pixels</td>
</tr>
<tr>
<td><strong>SetHeight (SizeT h)</strong></td>
<td>set render target height</td>
</tr>
<tr>
<td><strong>GetHeight () const</strong></td>
<td>get height of render target in pixels</td>
</tr>
<tr>
<td><strong>SetAntiAliasQuality (CoreGraphics::AntiAliasQuality::Code c)</strong></td>
<td>set antialias quality</td>
</tr>
<tr>
<td>CoreGraphics::AntiAliasQuality::Code</td>
<td>GetAntiAliasQuality ( ) const</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td>get anti-alias-quality</td>
</tr>
<tr>
<td>void</td>
<td>SetColorBufferFormat</td>
</tr>
<tr>
<td></td>
<td>(CoreGraphics::PixelFormat::Code colorFormat)</td>
</tr>
<tr>
<td></td>
<td>add a color buffer</td>
</tr>
<tr>
<td>CoreGraphics::PixelFormat::Code</td>
<td>GetColorBufferFormat ( ) const</td>
</tr>
<tr>
<td></td>
<td>get color buffer format at index</td>
</tr>
<tr>
<td>void</td>
<td>AddDepthStencilBuffer ( )</td>
</tr>
<tr>
<td></td>
<td>internally create a depth/stencil buffer</td>
</tr>
<tr>
<td>void</td>
<td>AddSharedDepthStencilBuffer (const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp;rt)</td>
</tr>
<tr>
<td></td>
<td>use external depth-stencil buffer</td>
</tr>
<tr>
<td>bool</td>
<td>HasDepthStencilBuffer ( ) const</td>
</tr>
<tr>
<td></td>
<td>return true if the render target has a depth/stencil buffer</td>
</tr>
<tr>
<td>void</td>
<td>SetMipMapsEnabled (bool b)</td>
</tr>
<tr>
<td></td>
<td>enable mipmap generation for this render target</td>
</tr>
<tr>
<td>bool</td>
<td>AreMipMapsEnabled ( ) const</td>
</tr>
<tr>
<td></td>
<td>get mipmap generation flag</td>
</tr>
<tr>
<td>void</td>
<td>SetResolveTextureResourceId (const Resources::ResourceId &amp;resId)</td>
</tr>
<tr>
<td></td>
<td>set resolve texture resource id</td>
</tr>
<tr>
<td>const Resources::ResourceId &amp;</td>
<td>GetResolveTextureResourceId ( ) const</td>
</tr>
<tr>
<td></td>
<td>get resolve texture resource id</td>
</tr>
<tr>
<td>void</td>
<td>SetResolveDepthTextureResourceId (const Resources::ResourceId &amp;resId)</td>
</tr>
<tr>
<td></td>
<td>set optional resolve depth texture resource id</td>
</tr>
<tr>
<td>const Resources::ResourceId &amp;</td>
<td>GetResolveDepthTextureResourceId ( ) const</td>
</tr>
<tr>
<td></td>
<td>get optional resolve depth texture resource id</td>
</tr>
<tr>
<td>void</td>
<td>SetResolveTextureWidth (SizeT w)</td>
</tr>
<tr>
<td></td>
<td>set resolve texture width</td>
</tr>
<tr>
<td>SizeT</td>
<td>GetResolveTextureWidth ( ) const</td>
</tr>
<tr>
<td></td>
<td>get resolve texture width</td>
</tr>
<tr>
<td>void</td>
<td>SetResolveTextureHeight (SizeT h)</td>
</tr>
</tbody>
</table>
set resolve texture height

**SizeT**  
`GetResolveTextureHeight () const`  
get resolve texture height

**void**  
`SetResolveTargetCpuAccess (bool b)`  
set cpu access flag

**bool**  
`GetResolveTargetCpuAccess () const`  
get cpu access flag

**void**  
`SetMRTIndex (IndexT i)`  
set optional MRT (Multiple Render Target) index, default is 0

**IndexT**  
`GetMRTIndex () const`  
get multiple-render-target index

**void**  
`SetClearFlags (uint clearFlags)`  
set clear flags

**uint**  
`GetClearFlags () const`  
get clear flags

**void**  
`SetClearColor (const Math::float4 &c)`  
set clear color

**const Math::float4 &**  
`GetClearColor () const`  
get clear color

**void**  
`SetClearDepth (float d)`  
set clear depth

**float**  
`GetClearDepth () const`  
get clear depth

**void**  
`SetClearStencil (uchar s)`  
set clear stencil value

**uchar**  
`GetClearStencil () const`  
get clear stencil value

**void**  
`SetResolveRect (const Math::rectangle<int>& r)`  
set the current resolve rectangle (in pixels)

**const Math::rectangle<int>&**  
`GetResolveRect () const`  
get resolve rectangle

**void**  
`BeginBatch (CoreGraphics::BatchType::Code batchType)`  
begin a batch
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void EndBatch()</code></td>
<td>end current batch</td>
</tr>
<tr>
<td><code>bool HasResolveTexture()</code></td>
<td>return true if resolve texture is valid</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::Texture &gt; &amp; GetResolveTexture()</code></td>
<td>get the resolve texture as Nebula texture object</td>
</tr>
<tr>
<td><code>bool HasCPUResolveTexture()</code></td>
<td>return true if cpu access resolve texture is valid</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::Texture &gt; &amp; GetCPUResolveTexture()</code></td>
<td>get the resolve texture as Nebula texture object</td>
</tr>
<tr>
<td><code>bool HasDepthResolveTexture()</code></td>
<td>return true if resolve texture is valid</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::Texture &gt; &amp; GetDepthResolveTexture()</code></td>
<td>get the resolve texture as Nebula texture object</td>
</tr>
<tr>
<td><code>virtual void ResolveDepthBuffer()</code></td>
<td>resolve depth buffer</td>
</tr>
<tr>
<td><code>SizeT GetMemorySize()</code></td>
<td>get byte size in memory, implemented in platform specific classes</td>
</tr>
<tr>
<td><code>void SetMemoryOffset(SizeT size)</code></td>
<td>set optional memory offset, not used by all platforms</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
void SetupMultiSampleType ()

setup compatible multisample type
```
Member Function Documentation

void Direct3D9::D3D9RenderTarget::SetupMultiSampleType ( ) [protected]

setup compatible multisample type

Select the antialias parameters that most closely resemble the preferred settings in the DisplayDevice object.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
Direct3D9::D3D9Shader
Direct3D9::D3D9Shader Class Reference

#include <d3d9shader.h>

Inheritance diagram for Direct3D9::D3D9Shader:
Detailed Description

D3D9 implementation of Shader.

Todo:
lost/reset device handling

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource states (DO NOT CHANGE ORDER!)</td>
<td></td>
</tr>
</tbody>
</table>
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9Shader ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~D3D9Shader ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>Unload ()</strong></td>
<td>Unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td>ID3DXEffect * <strong>GetD3D9Effect ()</strong></td>
<td>Get pointer to d3d effect</td>
</tr>
<tr>
<td><strong>CreateShaderInstance</strong></td>
<td>Create a shader instance from this shader</td>
</tr>
<tr>
<td><strong>DiscardShaderInstance</strong></td>
<td>Discard a shader instance</td>
</tr>
<tr>
<td>const <strong>GetAllShaderInstances</strong></td>
<td>Get all instances</td>
</tr>
<tr>
<td><strong>SetAsyncEnabled (bool b)</strong></td>
<td>Request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td><strong>IsAsyncEnabled ()</strong></td>
<td>Return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><strong>Lock ()</strong></td>
<td>Set locked to true</td>
</tr>
<tr>
<td><strong>Unlock ()</strong></td>
<td>Set locked to false</td>
</tr>
<tr>
<td><strong>IsLocked ()</strong></td>
<td>Returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td><strong>SetResourceId (ResourceId &amp;id)</strong></td>
<td>Set the resource identifier</td>
</tr>
<tr>
<td>const <strong>ResourceId &amp;</strong></td>
<td></td>
</tr>
</tbody>
</table>
GetResourceId () const

get the resource identifier

void SetLoader (const Ptr<
ResourceLoader> &loader)

set optional resource loader

const Ptr<
ResourceLoader> & GetLoader () const

get optional resource loader

void SetSaver (const
Ptr<
ResourceSaver> &saver)

set optional resource saver

const Ptr<
ResourceSaver> & GetSaver () const

get optional resource saver

SizeT GetUseCount () const

get current use count

virtual State Load ()

load the resource

void SetState (State s)

set current state (usually only called during Load()!)

State GetState () const

get current state

bool IsLoaded () const

return true if current state is Loaded

bool IsPending () const

return true if current state is Pending

bool LoadFailed () const

return true if current state is Failed

virtual bool Save ()

save the resource

int GetRefCount () const

get the current refcount

void AddRef ()

increment refcount by one

void Release ()

decrement refcount and destroy refcount is zero

bool IsInstanceOf (const
Rtti
const

return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const

return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const

return true if this object is instance of given class by fourcc

bool Isa (const Rtti &rtti) const

return true if this object is instance of given class, or a derived class

bool Isa (const Util::String &rttiName) const

return true if this object is instance of given class, or a derived class, by string

bool Isa (const Util::FourCC &rttiFourCC) const

return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const

get the class name

Util::FourCC GetClassFourCC () const

get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
<td><strong>IncrUseCount</strong>()</td>
<td>increment use count</td>
</tr>
<tr>
<td><code>void</code></td>
<td><strong>DecrUseCount</strong>()</td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Resource::State**
**Resources::Resource::Load** ( ) [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the **IsPending()** method will return true as long as the load is in progress, and **IsLoaded()** will become true when the loading process has finished. If the load has failed, **IsPending()** will switch to false and **IsLoaded()** will not be true.

**bool**
**Resources::Resource::Save** ( ) [virtual, inherited]

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

**int**
**Core::RefCounted::GetRefCount** ( ) const [inline, inherited]

going the current refcount

Return the current refcount of the object.

**void**
**Core::RefCounted::AddRef** ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

**void**
**Core::RefCounted::Release** ( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Direct3D9::D3D9ShaderInstance
Direct3D9::D3D9ShaderInstance Class Reference

#include <d3d9shaderinstance.h>

Inheritance diagram for Direct3D9::D3D9ShaderInstance:
Detailed Description

D3D9 implementation of CoreGraphics::ShaderInstance.

Todo:
lost/reset device handling

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>D3D9ShaderInstance()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~D3D9ShaderInstance()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>ID3DXEffect* GetD3D9Effect()</code> const</td>
<td>get pointer to d3d9 effect object</td>
</tr>
<tr>
<td><code>bool SelectActiveVariation</code> CoreGraphics::ShaderFeature::Mask featureMask`</td>
<td>select active variation by feature mask</td>
</tr>
<tr>
<td><code>SizeT Begin()</code></td>
<td>begin rendering through the currently selected variation, returns no. passes</td>
</tr>
<tr>
<td><code>void BeginPass(IndexT passIndex)</code></td>
<td>begin pass</td>
</tr>
<tr>
<td><code>void Commit()</code></td>
<td>commit changes before rendering</td>
</tr>
<tr>
<td><code>void EndPass()</code></td>
<td>end pass</td>
</tr>
<tr>
<td><code>void End()</code></td>
<td>end rendering through variation</td>
</tr>
<tr>
<td><code>void Discard()</code></td>
<td>discard the shader instance, must be called when no longer needed</td>
</tr>
<tr>
<td><code>bool IsValid()</code> const</td>
<td>return true if this object is valid</td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::Shader&gt; &amp; GetOriginalShader()</code> const</td>
<td>get pointer to original shader which created this instance</td>
</tr>
<tr>
<td><code>bool HasVariableByName(const CoreGraphics::ShaderVariable::Name &amp;n)</code> const</td>
<td>return true if the shader instance has a variable with the given name</td>
</tr>
<tr>
<td><code>bool HasVariableBySemantic</code> CoreGraphics::ShaderVariable::Semantic</td>
<td>return true if the shader instance has a variable with the given semantic</td>
</tr>
</tbody>
</table>
&n) const
return true if shader has variable by semantic

SizeT GetNumVariables() const
get number of variables

const Ptr<CoreGraphics::ShaderVariable> & GetVariableByIndex(IndexT i) const
get a variable by index

const Ptr<CoreGraphics::ShaderVariable> & GetVariableByName(const CoreGraphics::ShaderVariable::Name &n) const
get a variable by name

const Ptr<CoreGraphics::ShaderVariable> & GetVariableBySemantic(const CoreGraphics::ShaderVariable::Semantic &s) const
get a variable by semantic

bool HasVariation(CoreGraphics::ShaderFeature::Mask featureMask) const
return true if variation exists by matching feature mask

SizeT GetNumVariations() const
get number of variations in the shader

const Ptr<CoreGraphics::ShaderVariation> & GetVariationByIndex(IndexT i) const
get shader variation by index

const Ptr<CoreGraphics::ShaderVariation> & GetVariationByFeatureMask(CoreGraphics::ShaderFeature::Mask featureMask) const
get shader variation by feature mask

const Ptr<CoreGraphics::ShaderVariation> & GetActiveVariation() const
get currently active variation

int GetRefCount() const
get the current refcount

void AddRef()
increment refcount by one

void Release()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf(const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf(const Util::String &n) const
return true if shader has variable by semantic
<table>
<thead>
<tr>
<th>bool</th>
<th>&amp;className) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>virtual void</th>
<th><strong>Setup</strong> (const <em>Ptr&lt; CoreGraphics::Shader &gt;</em> &amp;origShader)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>setup the shader instance from its original shader object</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Cleanup</strong> ()</td>
</tr>
<tr>
<td></td>
<td>cleanup the shader instance</td>
</tr>
<tr>
<td>void</td>
<td><strong>OnLostDevice</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called by d3d9 shader server when d3d9 device is lost</td>
</tr>
<tr>
<td>void</td>
<td><strong>OnResetDevice</strong> ()</td>
</tr>
<tr>
<td></td>
<td>called by d3d9 shader server when d3d9 device is reset</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Direct3D9::D3D9ShaderInstance::Setup(const Ptr<CoreGraphics::Shader> origShader) [protected, virtual]

setup the shader instance from its original shader object

This method is called by Shader::CreateInstance() to setup the new shader instance.

Reimplemented from Base::ShaderInstanceBase.

void Base::ShaderInstanceBase::Discard() [inherited]

discard the shader instance, must be called when instance no longer needed

This method must be called when the object is no longer needed for proper cleanup.

int Core::RefCounted::GetRefCount() const [inline, inherited]

going the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrease refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Direct3D9::D3D9ShaderServer
Direct3D9::D3D9ShaderServer Class Reference

#include <d3d9shaderserver.h>

Inheritance diagram for Direct3D9::D3D9ShaderServer:
Detailed Description

D3D9 implementation of ShaderServer.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9ShaderServer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~D3D9ShaderServer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>bool <strong>Open ()</strong></td>
<td>Open the shader server</td>
</tr>
<tr>
<td>void <strong>Close ()</strong></td>
<td>Close the shader server</td>
</tr>
<tr>
<td>ID3DXEffectPool * <strong>GetD3D9EffectPool () const</strong></td>
<td>Get pointer to global effect pool</td>
</tr>
<tr>
<td>bool <strong>IsOpen () const</strong></td>
<td>Return true if the shader server is open</td>
</tr>
<tr>
<td>bool <strong>HasShader (const Resources::ResourceId &amp;resId)</strong></td>
<td>Return true if a shader exists</td>
</tr>
<tr>
<td><strong>Ptr&lt; CoreGraphics::ShaderInstance &gt; CreateShaderInstance (const Resources::ResourceId &amp;resId)</strong></td>
<td>Create a new shader instance</td>
</tr>
<tr>
<td><strong>GetAllShaders () const</strong></td>
<td>Get all loaded shaders</td>
</tr>
<tr>
<td><strong>SetActiveShaderInstance (const CoreGraphics::ShaderInstance &amp;shaderInst)</strong></td>
<td>Set currently active shader instance</td>
</tr>
<tr>
<td><strong>GetActiveShaderInstance () const</strong></td>
<td>Get currently active shader instance</td>
</tr>
<tr>
<td>void <strong>ResetFeatureBits ()</strong></td>
<td>Reset the current feature bits</td>
</tr>
<tr>
<td>void <strong>SetFeatureBits</strong></td>
<td>Set shader feature by bit mask</td>
</tr>
<tr>
<td>void <strong>ClearFeatureBits</strong></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>CoreGraphics::ShaderFeature::Mask</code> GetFeatureBits () const</td>
<td>get the current feature mask</td>
</tr>
<tr>
<td><code>CoreGraphics::ShaderFeature::Mask</code> FeatureStringToMask (const Util::String &amp;str)</td>
<td>convert a shader feature string into a feature bit mask</td>
</tr>
<tr>
<td><code>Util::String</code> FeatureMaskToString (CoreGraphics::ShaderFeature::Mask)</td>
<td>convert shader feature bit mask into string</td>
</tr>
<tr>
<td><code>void</code> ApplyObjectId (IndexT i)</td>
<td>apply an object id</td>
</tr>
<tr>
<td><code>bool</code> HasSharedVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &amp;sem) const</td>
<td>return true if a shared variable exists by semantic</td>
</tr>
<tr>
<td><code>SizeT</code> GetNumSharedVariables () const</td>
<td>get number of shared variables</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp;</code> GetSharedVariableByIndex (IndexT i) const</td>
<td>get a shared variable by index</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp;</code> GetSharedVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &amp;sem) const</td>
<td>get a shared variable by semantic</td>
</tr>
<tr>
<td><code>int</code> GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void</code> AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void</code> Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool</code> IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool</code> IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool</code> IsInstanceOf (const Util::FourCC &amp;fourcc) const</td>
<td>return true if this object is instance of given FourCC</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class or derived class.</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName)</code></td>
<td>Return true if this object is instance of given class or derived class, by string.</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFour)</code></td>
<td>Return true if this object is instance of given class or derived class, by fourcc.</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>Get the class name.</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>Get the class FourCC code.</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

`Ptr< ShaderInstance >`

Base::ShaderServerBase::CreateShaderInstance ( `Resources::ResourceId` `resId` ) [inherited]

create a new shader instance

This creates a clone of a template shader. This is the only method to create a new shader object. When the shader instance is no longer needed, call UnregisterShaderInstance() for proper cleanup.

`int`

Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

`void`

Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

`void`

Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

`const Util::String &`

Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

`Util::FourCC`
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Direct3D9::D3D9ShaderVariable
Direct3D9::D3D9ShaderVariable Class Reference

#include <d3d9shadervariable.h>

Inheritance diagram for Direct3D9::D3D9ShaderVariable:
Detailed Description

D3D9 implementation of ShaderVariable.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>enum</strong></td>
<td><strong>Type</strong></td>
</tr>
<tr>
<td></td>
<td><em>shader variable types</em></td>
</tr>
<tr>
<td><strong>typedef</strong></td>
<td><strong>Name</strong></td>
</tr>
<tr>
<td></td>
<td><em>shader variable name typedef</em></td>
</tr>
<tr>
<td><strong>typedef</strong></td>
<td><strong>Semantic</strong></td>
</tr>
<tr>
<td></td>
<td><em>shader variable semantic typedef</em></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3D9ShaderVariable ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~D3D9ShaderVariable ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>void SetInt (int value)</td>
<td>Set int value</td>
</tr>
<tr>
<td>void SetIntArray (const int *values, SizeT count)</td>
<td>Set int array values</td>
</tr>
<tr>
<td>void SetFloat (float value)</td>
<td>Set float value</td>
</tr>
<tr>
<td>void SetFloatArray (const float *values, SizeT count)</td>
<td>Set float array values</td>
</tr>
<tr>
<td>void SetFloat4 (const Math::float4 &amp;value)</td>
<td>Set vector value</td>
</tr>
<tr>
<td>void SetFloat4Array (const Math::float4 *values, SizeT count)</td>
<td>Set vector array values</td>
</tr>
<tr>
<td>void SetMatrix (const Math::matrix44 &amp;value)</td>
<td>Set matrix value</td>
</tr>
<tr>
<td>void SetMatrixArray (const Math::matrix44 *values, SizeT count)</td>
<td>Set matrix array values</td>
</tr>
<tr>
<td>void SetBool (bool value)</td>
<td>Set bool value</td>
</tr>
<tr>
<td>void SetBoolArray (const bool *values, SizeT count)</td>
<td>Set bool array values</td>
</tr>
</tbody>
</table>
void SetTexture (const Ptr<CoreGraphics::Texture> &value)
set texture value

Ptr< CoreGraphics::ShaderVariableInstance > CreateInstance ()
create a shader variable instance

Type GetType () const
get the data type of the variable

const Name & GetName () const
get the name of the variable

const Semantic & GetSemantic () const
get the semantics of the variable

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC &amp; GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Util::String</code></td>
<td><strong>TypeToString</strong> <em>(Type t)</em></td>
<td>convert type to string</td>
</tr>
<tr>
<td><code>Type</code></td>
<td><strong>StringToType</strong> <em>(const Util::String &amp;str)</em></td>
<td>convert string to type</td>
</tr>
<tr>
<td><code>void</code></td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application <em>(NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetType(Type t)</code></td>
<td>set variable type</td>
</tr>
<tr>
<td><code>voidSetName(const Name &amp;n)</code></td>
<td>set variable name</td>
</tr>
<tr>
<td><code>void SetSemantic(const Semantic &amp;s)</code></td>
<td>set variable semantic</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Direct3D9::D3D9ShaderVariation
Direct3D9::D3D9ShaderVariation Class Reference

#include <d3d9shadervariation.h>

Inheritance diagram for Direct3D9::D3D9ShaderVariation:
Detailed Description

Under Direct3D9, a shader variation is represented by an d3dx effect technique which must be annotated by a FeatureMask string.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9ShaderVariation ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~D3D9ShaderVariation ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>D3DXHANDLE GetD3D9Technique () const</strong></td>
<td>get the D3DX technique handle</td>
</tr>
<tr>
<td><strong>ID3DXEffect * GetD3D9Effect () const</strong></td>
<td>get the D3DX effect which owns this variation</td>
</tr>
<tr>
<td><strong>const Name &amp; GetName () const</strong></td>
<td>get the shader variation's name</td>
</tr>
<tr>
<td><strong>CoreGraphics::ShaderFeature::Mask GetFeatureMask () const</strong></td>
<td>get the feature bit mask of this variation</td>
</tr>
<tr>
<td><strong>SizeT GetNumPasses () const</strong></td>
<td>get number of passes in this variation</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>bool IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetName (const Name &amp;n)</code></td>
<td>set variation name</td>
</tr>
<tr>
<td><code>SetFeatureMask (CoreGraphics::ShaderFeature::Mask m)</code></td>
<td>set feature bit mask of this variation</td>
</tr>
<tr>
<td><code>SetNumPasses (SizeT n)</code></td>
<td>set number of passes</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
```
builds only!

This method should be called as the very last before an application exits.
Direct3D9::D3D9StreamShaderLoader
Direct3D9::D3D9StreamShaderLoader
Class Reference

#include <d3d9streamshaderloader.h>

Inheritance diagram for Direct3D9::D3D9StreamShaderLoader:
Detailed Description

D3D9 implementation of StreamShaderLoader.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CanLoadAsync()</code> const</td>
<td>return true if asynchronous loading is supported</td>
</tr>
<tr>
<td><code>OnLoadRequested()</code></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td><code>OnLoadCancelled()</code></td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td><code>OnPending()</code></td>
<td>call frequently while after <code>OnLoadRequested()</code> to put Resource into loaded state</td>
</tr>
<tr>
<td><code>OnAttachToResource</code> (const <code>Ptr&lt; Resource &gt;</code> &amp;res)</td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td><code>OnRemoveFromResource()</code></td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td><code>IsAttachedToResource()</code> const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td><code>GetResource()</code> const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td><code>GetState()</code> const</td>
<td>return current state</td>
</tr>
<tr>
<td><code>Reset()</code></td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td><code>GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf</code> (const <code>Util::String</code> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>
**bool IsInstanceOf** (const Util::FourCC &classFourCC) const

*return true if this object is instance of given class by fourcc*

**bool IsA** (const Rtti &rtti) const

*return true if this object is instance of given class, or a derived class*

**bool IsA** (const Util::String &rttiName) const

*return true if this object is instance of given class, or a derived class, by string*

**bool IsA** (const Util::FourCC &rttiFourCC) const

*return true if this object is instance of given class, or a derived class, by fourcc*

**const Util::String & GetClassName** () const

*get the class name*

**Util::FourCC GetClassFourCC** () const

*get the class FourCC code*
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

```c++
void SetState (Resource::State S)

set current state
```
**Member Function Documentation**

```cpp
bool Resources::StreamResourceLoader::OnLoadRequested() [virtual, inherited]
called by resource when a load is requested

Handle the generic load request. In the asynchronous case, this method will fire a ReadStream message and go into pending state. The client will then call `OnPending()` periodically which checks whether the ReadStream message has been handled and continue accordingly. In the synchronous state the method will create an `IO::Stream` object and call the `SetupResourceFromStream()` method directly.

Reimplemented from `Resources::ResourceLoader`.

Reimplemented in `Resources::D3D9TextureStreamer`.
```
```cpp
void Resources::StreamResourceLoader::OnLoadCancelled() [virtual, inherited]
called by resource to cancel a pending load

This method is called when the currently pending asynchronous load request should be cancelled.

Reimplemented from `Resources::ResourceLoader`.
```
```cpp
bool Resources::StreamResourceLoader::OnPending() [virtual, inherited]
call frequently while after `OnLoadRequested()` to put `Resource` into loaded state

This method is called periodically by the client when the resource is in pending state. The pending ReadStream message will be checked, and if it has been handled successfully the `SetupResourceFromStream()` method will be called and the
Resource will go into Loaded state. If anything goes wrong, the resource will go into Failed state.

Reimplemented from Resources::ResourceLoader.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Direct3D9::D3D9StreamTextureLoader
Direct3D9::D3D9StreamTextureLoader
Class Reference

#include <d3d9streamtextureloader.h>

Inheritance diagram for Direct3D9::D3D9StreamTextureLoader:
Detailed Description

D3D9/Xbox360 implementation of StreamTextureLoader.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool CanLoadAsync () const</td>
<td>override in subclass: return true if asynchronous loading is supported (default is true)</td>
</tr>
<tr>
<td>virtual bool OnLoadRequested ()</td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td>virtual void OnLoadCancelled ()</td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td>virtual void OnPending ()</td>
<td>call frequently while after OnLoadRequested() to put Resource into loaded state</td>
</tr>
<tr>
<td>virtual void OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>virtual void OnRemoveFromResource ()</td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>bool IsAttachedToResource () const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetResource () const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td>Resource::State GetState () const</td>
<td>return current state</td>
</tr>
<tr>
<td>virtual void Reset ()</td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
void SetState (Resource::State S)
```

set current state
Member Function Documentation

```cpp
bool Resources::StreamResourceLoader::CanLoadAsync() const [virtual, inherited]
```
override in subclass: return true if asynchronous loading is supported (default is true)

Indicate whether this resource loader supports asynchronous loading. The default is true. Override this method in a subclass and return false otherwise.

Reimplemented from `Resources::ResourceLoader`.
Reimplemented in `Direct3D9::D3D9StreamShaderLoader`.

```cpp
bool Resources::StreamResourceLoader::OnLoadRequested() [virtual, inherited]
```
called by resource when a load is requested

Handle the generic load request. In the asynchronous case, this method will fire a ReadStream message and go into pending state. The client will then call `OnPending()` periodically which checks whether the ReadStream message has been handled and continue accordingly. In the synchronous state the method will create an `IO::Stream` object and call the `SetupResourceFromStream()` method directly.

Reimplemented from `Resources::ResourceLoader`.
Reimplemented in `Resources::D3D9TextureStreamer`.

```cpp
void Resources::StreamResourceLoader::OnLoadCancelled() [virtual, inherited]
```
called by resource to cancel a pending load

This method is called when the currently pending asynchronous load
request should be cancelled.

Reimplemented from `Resources::ResourceLoader`.

```cpp
bool
Resources::StreamResourceLoader::OnPending( ) [virtual, inherited]
```

call frequently while after `OnLoadRequested()` to put `Resource` into loaded state

This method is called periodically by the client when the resource is in pending state. The pending ReadStream message will be checked, and if it has been handled successfully the `SetupResourceFromStream()` method will be called and the `Resource` will go into Loaded state. If anything goes wrong, the resource will go into Failed state.

Reimplemented from `Resources::ResourceLoader`.

```cpp
int
Core::RefCounted::GetRefCount( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void
Core::RefCounted::AddRef( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]
```
get the class name

Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Direct3D9::D3D9TextRenderer
Direct3D9::D3D9TextRenderer Class Reference

#include <d3d9textrenderer.h>

Inheritance diagram for Direct3D9::D3D9TextRenderer:
Detailed Description

Implements a simple text renderer for Direct3D9. This is only intended for outputting debug text, not for high-quality text rendering!

FIXME: Need to handle Lost Device (ID3DXFont)

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9TextRenderer</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~D3D9TextRenderer</strong> ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>void <strong>Open</strong> ()</td>
<td>Open the device</td>
</tr>
<tr>
<td>void <strong>Close</strong> ()</td>
<td>Close the device</td>
</tr>
<tr>
<td>void <strong>DrawTextElements</strong> ()</td>
<td>Draw the accumulated text</td>
</tr>
<tr>
<td>bool <strong>IsOpen</strong> () const</td>
<td>Check if text renderer open</td>
</tr>
<tr>
<td>void <strong>DeleteTextElementsByThreadId</strong> (Threading::ThreadId threadId)</td>
<td>Delete added text by thread ID</td>
</tr>
<tr>
<td>void <strong>AddTextElement</strong> (const CoreGraphics::TextElement &amp;textElement)</td>
<td>Add text element for rendering</td>
</tr>
<tr>
<td>void <strong>AddTextElements</strong> (const Util::Array&lt;CoreGraphics::TextElement &gt; &amp;textElement)</td>
<td>Add multiple text elements for rendering</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>
| bool **IsInstanceOf** (const Util::FourCC &classFourCC) | }
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><strong>const</strong> return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const</td>
<td><strong>Util::String &amp; GetClassName() const</strong> get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC () const</strong> get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Direct3D9::D3D9TextRenderer::DrawTextElements()

draw the accumulated text

Draw buffered text. This method is called once per frame.

Reimplemented from `Base::TextRendererBase`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Direct3D9::D3D9Texture
Direct3D9::D3D9Texture Class Reference

#include <d3d9texture.h>

Inheritance diagram for Direct3D9::D3D9Texture:
Detailed Description

D3D9/Xbox360 implementation of Texture class

FIXME: need to handle DeviceLost through RenderDevice event handler \(\text{(Win32 only)}\)

(C) 2007 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>Enum</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Texture types</td>
<td></td>
</tr>
<tr>
<td>Enum</td>
<td>CubeFace</td>
<td>Cube map face</td>
</tr>
<tr>
<td>Enum</td>
<td>Usage</td>
<td>Resource usage flags</td>
</tr>
<tr>
<td>Enum</td>
<td>State</td>
<td>Resource states</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9Texture()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~D3D9Texture()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void Unload()</td>
<td>Unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td>bool Map(IndexT mipLevel, MapType mapType, MapInfo &amp;outMapInfo)</td>
<td>Map a texture mip level for CPU access</td>
</tr>
<tr>
<td>void Unmap(IndexT mipLevel)</td>
<td>Unmap texture after CPU access</td>
</tr>
<tr>
<td>bool MapCubeFace(CubeFace face, IndexT mipLevel, MapType mapType, MapInfo &amp;outMapInfo)</td>
<td>Map a cube map face for CPU access</td>
</tr>
<tr>
<td>void UnmapCubeFace(CubeFace face, IndexT mipLevel)</td>
<td>Unmap cube map face after CPU access</td>
</tr>
<tr>
<td>IDirect3DBaseTexture9 * GetD3D9BaseTexture() const</td>
<td>Get D3D9 base texture pointer</td>
</tr>
<tr>
<td>IDirect3DTexture9 * GetD3D9Texture() const</td>
<td>Get D3D9 texture pointer</td>
</tr>
<tr>
<td>IDirect3DCubeTexture9 * GetD3D9CubeTexture() const</td>
<td>Get D3D9 cube texture pointer</td>
</tr>
<tr>
<td>IDirect3DVolumeTexture9 * GetD3D9VolumeTexture() const</td>
<td>Get D3D9 volume texture pointer</td>
</tr>
<tr>
<td>void SetupFromD3D9Texture(IDirect3DTexture9 *ptr, const bool setLoaded=true)</td>
<td>Setup from a IDirect3DTexture9</td>
</tr>
<tr>
<td>void SetupFromD3D9CubeTexture(IDirect3DCubeTexture9 *ptr, const bool setLoaded=true)</td>
<td>Setup from a IDirect3DCubeTexture</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>SetupFromD3D9VolumeTexture</code></td>
<td>Setup from a IDirect3DVolumeTexture</td>
</tr>
<tr>
<td><code>GetType()</code></td>
<td>Get texture type</td>
</tr>
<tr>
<td><code>GetWidth()</code></td>
<td>Get width of texture</td>
</tr>
<tr>
<td><code>GetHeight()</code></td>
<td>Get height of texture (if 2d or 3d texture)</td>
</tr>
<tr>
<td><code>GetDepth()</code></td>
<td>Get depth of texture (if 3d texture)</td>
</tr>
<tr>
<td><code>GetNumMipLevels()</code></td>
<td>Get number of mip levels</td>
</tr>
<tr>
<td><code>GetSkippedMips()</code></td>
<td>Get number of currently skipped mip levels</td>
</tr>
<tr>
<td><code>SetSkippedMips(SizeT m)</code></td>
<td>Set number of currently skipped mip levels</td>
</tr>
<tr>
<td><code>GetPixelFormat()</code></td>
<td>Get pixel format of the texture</td>
</tr>
<tr>
<td><code>Map(IndexT mipLevel, Access accessMode, MapInfo &amp;outMapInfo)</code></td>
<td>Map the a texture mip level for CPU access</td>
</tr>
<tr>
<td><code>MapCubeFace(CubeFace face, IndexT mipLevel, Access accessMode, MapInfo &amp;outMapInfo)</code></td>
<td>Map a cube map face for CPU access</td>
</tr>
<tr>
<td><code>SetUsage(Usage usage)</code></td>
<td>Set resource usage type</td>
</tr>
<tr>
<td><code>GetUsage()</code></td>
<td>Get resource usage type</td>
</tr>
<tr>
<td><code>SetAccess(Access access)</code></td>
<td>Set resource cpu access type</td>
</tr>
<tr>
<td><code>GetAccess()</code></td>
<td>Get cpu access type</td>
</tr>
<tr>
<td><code>SetAsyncEnabled(bool b)</code></td>
<td>Request synchronous/asynchronous resource</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsAsyncEnabled()</code></td>
<td>const; return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>void Lock()</code></td>
<td>set locked to true</td>
</tr>
<tr>
<td><code>void Unlock()</code></td>
<td>set locked to false</td>
</tr>
<tr>
<td><code>bool IsLocked()</code></td>
<td>const; returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td><code>void SetResourceID(const ResourceID &amp;id)</code></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><code>const ResourceID &amp; GetResourceID()</code></td>
<td>get the resource identifier</td>
</tr>
<tr>
<td><code>void SetLoader(const Ptr&lt;ResourceLoader&gt; &amp;loader)</code></td>
<td>set optional resource loader</td>
</tr>
<tr>
<td><code>const Ptr&lt;ResourceLoader&gt; &amp; GetLoader()</code></td>
<td>get optional resource loader</td>
</tr>
<tr>
<td><code>void SetSaver(const Ptr&lt;ResourceSaver&gt; &amp;saver)</code></td>
<td>set optional resource saver</td>
</tr>
<tr>
<td><code>const Ptr&lt;ResourceSaver&gt; &amp; GetSaver()</code></td>
<td>get optional resource saver</td>
</tr>
<tr>
<td><code>SizeT GetUseCount()</code></td>
<td>get current use count</td>
</tr>
<tr>
<td><code>virtual State Load()</code></td>
<td>load the resource</td>
</tr>
<tr>
<td><code>void SetState(State s)</code></td>
<td>set current state (usually only called during Load()!)</td>
</tr>
<tr>
<td><code>State GetState()</code></td>
<td>get current state</td>
</tr>
<tr>
<td><code>bool IsLoaded()</code></td>
<td>const; return true if current state is Loaded</td>
</tr>
<tr>
<td><code>bool IsPending()</code></td>
<td>const; return true if current state is Pending</td>
</tr>
<tr>
<td>C++ Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>bool LoadFailed () const</code></td>
<td>return true if current state is Failed</td>
</tr>
<tr>
<td>virtual <code>bool Save ()</code></td>
<td>save the resource</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void DumpRefCountingLeaks ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetType (Type t)</code></td>
<td>set texture type</td>
</tr>
<tr>
<td><code>void SetWidth (SizeT w)</code></td>
<td>set texture width</td>
</tr>
<tr>
<td><code>void SetHeight (SizeT h)</code></td>
<td>set texture height</td>
</tr>
<tr>
<td><code>void SetDepth (SizeT d)</code></td>
<td>set texture depth</td>
</tr>
<tr>
<td><code>void SetNumMipLevels (SizeT n)</code></td>
<td>set number of mip levels</td>
</tr>
<tr>
<td><code>void SetPixelFormat (CoreGraphics::PixelFormat::Code f)</code></td>
<td>set pixel format</td>
</tr>
<tr>
<td><code>void IncrUseCount ()</code></td>
<td>increment use count</td>
</tr>
<tr>
<td><code>void DecrUseCount ()</code></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Direct3D9::D3D9Texture::SetupFromD3D9Texture ( *tex2D, const bool setLoaded = true )

setup from a IDirect3DTexture9

Helper method to setup the texture object from a D3D9 2D texture.

```cpp
void Direct3D9::D3D9Texture::SetupFromD3D9CubeTexture ( *texCube, const bool setLoaded = true )

setup from a IDirect3DCubeTexture

Helper method to setup the texture object from a D3D9 cube texture.

```cpp
void Direct3D9::D3D9Texture::SetupFromD3D9VolumeTexture ( *texVolume, const bool setLoaded = true )

setup from a IDirect3DVolumeTexture

Helper method to setup the texture object from a D3D9 volume texture.

```cpp
Resource::State Resources::Resource::Load ( ) [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded
asynchronously, the \texttt{IsPending()} method will return true as long as the load is in progress, and \texttt{IsLoaded()} will become true when the loading process has finished. If the load has failed, \texttt{IsPending()} will switch to false and \texttt{IsLoaded()} will not be true.

\begin{verbatim}
bool
Resources::Resource::Save ( ) [virtual, inherited]

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

\end{verbatim}

\begin{verbatim}
int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

\end{verbatim}

\begin{verbatim}
void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

\end{verbatim}

\begin{verbatim}
void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

\end{verbatim}

\begin{verbatim}
const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.
\end{verbatim}
Util::FourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:43 2010
Frame::FrameBatch
Frame::FrameBatch Class Reference

#include <framebatch.h>

Inheritance diagram for Frame::FrameBatch:
Detailed Description

A frame batch encapsulates the rendering of a batch of ModelNodeInstances.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FrameBatch ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~FrameBatch ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>Discard the frame batch</td>
</tr>
<tr>
<td>void Render ()</td>
<td>Render the batch</td>
</tr>
<tr>
<td>void SetShader (const Ptr&lt; CoreGraphics::ShaderInstance&gt; &amp; )</td>
<td>Set batch shader</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp; GetShader () const</td>
<td>Get batch shader</td>
</tr>
<tr>
<td>void SetType (CoreGraphics::BatchType::Code )</td>
<td>Set batch type</td>
</tr>
<tr>
<td>CoreGraphics::BatchType::Code GetType () const</td>
<td>Get batch type</td>
</tr>
<tr>
<td>void SetNodeFilter (Models::ModelNodeType::Code )</td>
<td>Set model node filter</td>
</tr>
<tr>
<td>Models::ModelNodeType::Code GetNodeFilter () const</td>
<td>Get model node filter</td>
</tr>
<tr>
<td>void SetLightingMode (LightingMode::Code )</td>
<td>Set lighting mode</td>
</tr>
<tr>
<td>LightingMode::Code GetLightingMode () const</td>
<td>Get lighting mode</td>
</tr>
<tr>
<td>void SetSortingMode (SortingMode::Code )</td>
<td>Set sorting mode</td>
</tr>
<tr>
<td>SortingMode::Code GetSortingMode () const</td>
<td>Get sorting mode</td>
</tr>
</tbody>
</table>
void SetShaderFeatures (CoreGraphics::ShaderFeature::Mask m)  
set shader features

CoreGraphics::ShaderFeature::Mask GetShaderFeatures () const  
get shader features

void AddVariable (const Ptr<CoreGraphics::ShaderVariableInstance> &var)  
add a shader variable instance to the frame batch

SizeT GetNumVariables () const  
get number of shader variables

const Ptr<CoreGraphics::ShaderVariableInstance> & GetVariableByIndex (IndexT i) const  
get shader variable by index

int GetRefCount () const  
get the current refcount

void AddRef ()  
increment refcount by one

void Release ()  
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const  
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const  
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const  
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const  
return true if this object is instance of given derived class

bool IsA (const Util::String &rttiName) const  
return true if this object is instance of given derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const  
return true if this object is instance of given derived class, by fourcc
<table>
<thead>
<tr>
<th>const</th>
<th>return true if this object is instance of a derived class, by fourcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

int Core::RefCounted::GetRefCount(const [inline, inherited])
get the current refcount
Return the current refcount of the object.

void Core::RefCounted::AddRef( ) [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void Core::RefCounted::Release( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
get the class name
Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Frame::FramePass
Frame::FramePass Class Reference

#include <framepass.h>

Inheritance diagram for Frame::FramePass:
Detailed Description

A frame pass encapsulates all 3d rendering to a render target, organized into FrameBatches.

(C) 2009 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FramePass</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~<strong>FramePass</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <strong>Discard</strong> ()</td>
<td>discard the frame pass</td>
</tr>
<tr>
<td>virtual void <strong>Render</strong> ()</td>
<td>render the pass</td>
</tr>
<tr>
<td>void <strong>SetName</strong> (const Resources::ResourceId &amp;resId)</td>
<td>set the name of the frame pass</td>
</tr>
<tr>
<td>const Resources::ResourceId &amp; <strong>GetName</strong> () const</td>
<td>get the name of the frame pass</td>
</tr>
<tr>
<td>void <strong>SetShader</strong> (const Ptr&lt; CoreGraphics::ShaderInstance &amp;&gt; &amp;var)</td>
<td>set pass shader</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp; <strong>GetShader</strong> () const</td>
<td>get pass shader</td>
</tr>
<tr>
<td>void <strong>SetRenderTarget</strong> (const Ptr&lt; CoreGraphics::RenderTarget &amp;&gt; &amp;rt)</td>
<td>set render target</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; <strong>GetRenderTarget</strong> () const</td>
<td>get render target</td>
</tr>
<tr>
<td>void <strong>SetMultipleRenderTarget</strong> (const CoreGraphics::MultipleRenderTarget &amp;rt)</td>
<td>set multiple render target</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::MultipleRenderTarget &gt; &amp; <strong>GetMultipleRenderTarget</strong> () const</td>
<td>get multiple render target</td>
</tr>
<tr>
<td>void <strong>AddVariable</strong> (const Ptr&lt; CoreGraphics::ShaderVariableInstance &gt; &amp;var)</td>
<td>add a shader variable instance to the frame pass</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SizeT GetNumVariables () const</td>
<td>get number of shader variables</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::ShaderVariableInstance &gt; &amp; GetVariableByIndex (IndexT i) const</td>
<td>get shader variable by index</td>
</tr>
<tr>
<td>void AddBatch (const Ptr&lt; FrameBatch &gt;&amp; batch)</td>
<td>add a frame batch to the frame pass</td>
</tr>
<tr>
<td>SizeT GetNumBatches () const</td>
<td>get number of frame batches</td>
</tr>
<tr>
<td>const Ptr&lt; FrameBatch &gt; &amp; GetBatchByIndex (IndexT i) const</td>
<td>get batch by index</td>
</tr>
<tr>
<td>void SetClearColor (const Math::float4&amp; clearColor)</td>
<td>set clear color for all rendertargets</td>
</tr>
<tr>
<td>const Math::float4 &amp; GetClearColor () const</td>
<td>get clear color</td>
</tr>
<tr>
<td>void SetClearDepth (float d)</td>
<td>set clear depth for all rendertargets</td>
</tr>
<tr>
<td>float GetClearDepth () const</td>
<td>get clear depth</td>
</tr>
<tr>
<td>void SetClearStencil (uchar s)</td>
<td>set clear stencil value for all rendertargets</td>
</tr>
<tr>
<td>uchar GetClearStencil () const</td>
<td>get clear stencil value</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti)</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;str)</td>
<td>return true if this string is equal to given string</td>
</tr>
<tr>
<td>C++ Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount
Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name
Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
```
builds only!

This method should be called as the very last before an application exits.
Frame::FramePassBase
Frame::FramePassBase Class Reference

#include <framepassbase.h>

Inheritance diagram for Frame::FramePassBase:
Detailed Description

A frame pass encapsulates all 3d rendering to a render target, organized into FrameBatches.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FramePassBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~FramePassBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>Discard ()</strong></td>
<td>Discard the frame pass</td>
</tr>
<tr>
<td>virtual void <strong>Render ()</strong></td>
<td>Render the pass</td>
</tr>
<tr>
<td><strong>SetName (const Resources::ResourceId &amp;resId)</strong></td>
<td>Set the name of the frame pass</td>
</tr>
<tr>
<td>const Resources::ResourceId &amp; <strong>GetName () const</strong></td>
<td>Get the name of the frame pass</td>
</tr>
<tr>
<td>void <strong>SetShader (const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp;inst)</strong></td>
<td>Set pass shader</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp; <strong>GetShader () const</strong></td>
<td>Get pass shader</td>
</tr>
<tr>
<td>void <strong>SetRenderTarget (const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp;target)</strong></td>
<td>Set render target</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; <strong>GetRenderTarget () const</strong></td>
<td>Get render target</td>
</tr>
<tr>
<td>void <strong>SetMultipleRenderTarget (const CoreGraphics::MultipleRenderTarget &amp;rt)</strong></td>
<td>Set multiple render target</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::MultipleRenderTarget &gt; &amp; <strong>GetMultipleRenderTarget () const</strong></td>
<td>Get multiple render target</td>
</tr>
<tr>
<td>void <strong>AddVariable (const Ptr&lt; CoreGraphics::ShaderVariableInstance &gt; &amp;var)</strong></td>
<td>Add a shader variable instance to the frame pass</td>
</tr>
</tbody>
</table>
GetSizeT() const
get number of shader variables

const Ptr<CoreGraphics::ShaderVariableInstance> &
GetVariableByIndex(IndexT i) const
get shader variable by index

void AddBatch(const Ptr<FrameBatch> &batch)
add a frame batch to the frame pass

GetSizeT() const
get number of frame batches

const Ptr<FrameBatch> &
GetBatchByIndex(IndexT i) const
get batch by index

void SetClearFlags(uint clearFlags)
set clear flags (mask of RenderTarget::Clear*)

uint GetClearFlags() const
get clear flags

void SetClearColor(const Math::float4 &)
set clear color for all render targets

const Math::float4 &
GetClearColor() const
get clear color

void SetClearDepth(float d)
set clear depth for all render targets

float GetClearDepth() const
get clear depth

void SetClearStencil(uchar s)
set clear stencil value for all render targets

uchar GetClearStencil() const
get clear stencil value

int GetRefCount() const
get the current refcount

void AddRef()
increment refcount by one

void Release()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf(const Rtti &rtti) const
return true if this object is instance of given class

IsInstanceOf(const Util::String &str) const
return true if this object is instance of given string
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class or derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class or derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class or derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Frame::FramePostEffect
Frame::FramePostEffect Class Reference

#include <frameposteffect.h>

Inheritance diagram for Frame::FramePostEffect:
Detailed Description

A frame post-effect implements draws a fullscreen quad through a shader which implements the post effect. Additionally it is possible to add render batches to a post effect, these will be called after the fullscreen quad is rendered, allowing the application to render stuff on top of the final frame.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FramePostEffect()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~FramePostEffect()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <code>Setup()</code></td>
<td>Setup the post effect</td>
</tr>
<tr>
<td>virtual void <code>Discard()</code></td>
<td>Discard the post effect</td>
</tr>
<tr>
<td>virtual void <code>Render()</code></td>
<td>Render the post effect</td>
</tr>
<tr>
<td>void <code>SetName(const Resources::ResourceId &amp;)</code></td>
<td>Set the name of the frame pass</td>
</tr>
<tr>
<td>const Resources::ResourceId &amp; <code>GetName()</code></td>
<td>Get the name of the frame pass</td>
</tr>
<tr>
<td>void <code>SetShader(const CoreGraphics::ShaderInstance &amp;)</code></td>
<td>Set pass shader</td>
</tr>
<tr>
<td>const CoreGraphics::ShaderInstance &amp; <code>GetShader()</code></td>
<td>Get pass shader</td>
</tr>
<tr>
<td>void <code>SetRenderTarget(const CoreGraphics::RenderTarget &amp;)</code></td>
<td>Set render target</td>
</tr>
<tr>
<td>const CoreGraphics::RenderTarget &amp; <code>GetRenderTarget()</code></td>
<td>Get render target</td>
</tr>
<tr>
<td>void <code>SetMultipleRenderTarget(const CoreGraphics::MultipleRenderTarget &amp;)</code></td>
<td>Set multiple render target</td>
</tr>
<tr>
<td>const CoreGraphics::MultipleRenderTarget &amp; <code>GetMultipleRenderTarget()</code></td>
<td>Get multiple render target</td>
</tr>
<tr>
<td>void <code>AddVariable(const CoreGraphics::ShaderVariableInstance &amp;)</code></td>
<td>Add variable</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>add a shader variable instance to the frame pass</td>
<td><code>GetNumVariables()</code> const SizeT</td>
</tr>
<tr>
<td>get number of shader variables</td>
<td><code>GetVariableByIndex()</code> (IndexT i) const</td>
</tr>
<tr>
<td>get shader variable by index</td>
<td><code>AddBatch()</code> (const Ptr&lt; FrameBatch &gt; &amp;batch) add a frame batch to the frame pass</td>
</tr>
<tr>
<td>get number of frame batches</td>
<td><code>GetNumBatches()</code> const SizeT</td>
</tr>
<tr>
<td>get batch by index</td>
<td><code>GetBatchByIndex()</code> (IndexT i) const</td>
</tr>
<tr>
<td>set clear flags (mask ofRenderTarget::Clear*)</td>
<td><code>SetClearFlags()</code> (uint clearFlags)</td>
</tr>
<tr>
<td>get clear flags</td>
<td><code>GetClearFlags()</code> const uint</td>
</tr>
<tr>
<td>set clear color for all rendertargets</td>
<td><code>SetClearColor()</code> (const Math::float4 &amp;clearColor) set clear color</td>
</tr>
<tr>
<td>get clear color</td>
<td><code>GetClearColor()</code> const Math::float4 &amp;</td>
</tr>
<tr>
<td>set clear depth for all rendertargets</td>
<td><code>SetClearDepth()</code> (float d) set clear depth</td>
</tr>
<tr>
<td>get clear depth</td>
<td><code>GetClearDepth()</code> const float</td>
</tr>
<tr>
<td>set clear stencil value for all rendertargets</td>
<td><code>SetClearStencil()</code> (uchar s) set clear stencil value</td>
</tr>
<tr>
<td>get clear stencil value</td>
<td><code>GetClearStencil()</code> const uchar</td>
</tr>
<tr>
<td>get the current refcount</td>
<td><code>GetRefCount()</code> const int</td>
</tr>
<tr>
<td>increment refcount by one</td>
<td><code>AddRef()</code></td>
</tr>
<tr>
<td>decrement refcount and destroy object if refcount is zero</td>
<td><code>Release()</code></td>
</tr>
<tr>
<td>IsInstanceOf() (const Rtti &amp;rtti)</td>
<td>bool</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;className) const</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td>const Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td>static void</td>
<td>DumpRefCountingLeaks ()</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Frame::FramePostEffect::Setup()

setup the post effect

This sets up the quad mesh for rendering the fullscreen quad.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current ref count

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment ref count by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement ref count and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Frame::FrameServer
Frame::FrameServer Class Reference

#include <frameserver.h>

Inheritance diagram for Frame::FrameServer:
Detailed Description

Server object of the frame subsystem. Factory for FrameShaders.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FrameServer ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~FrameServer ()</td>
<td>destructor</td>
</tr>
<tr>
<td>bool Open ()</td>
<td>open the frame server (loads all frame shaders)</td>
</tr>
<tr>
<td>void Close ()</td>
<td>close the frame server</td>
</tr>
<tr>
<td>bool IsOpen ()</td>
<td>return true if open</td>
</tr>
<tr>
<td>Ptr&lt; FrameShader &gt;</td>
<td>LookupFrameShader (const Resources::ResourceId &amp;name) gain access to a frame shader by name, shader will be loaded if not happened yet</td>
</tr>
<tr>
<td>int GetRefCount ()</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti)</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className)</td>
<td>return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti)</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName)</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC)</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong> <strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```c++
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```c++
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```c++
void Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```c++
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```c++
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```c++
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG}
builds only!

This method should be called as the very last before an application exits.
Frame::FrameShader
Frame::FrameShader Class Reference

#include <frameshader.h>

Inheritance diagram for Frame::FrameShader:
Detailed Description

A **FrameShader** controls the rendering of an entire frame, and is configured by an XML file.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FrameShader()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~FrameShader()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Discard()</strong></td>
<td>discard the frame shader</td>
</tr>
<tr>
<td><strong>void Render()</strong></td>
<td>render the frame shader from the given camera</td>
</tr>
<tr>
<td><strong>void SetName(const Resources::ResourceId &amp;id)</strong></td>
<td>set the name of the frame shader</td>
</tr>
<tr>
<td><strong>const Resources::ResourceId &amp; GetName() const</strong></td>
<td>get the name of the frame shader</td>
</tr>
<tr>
<td><strong>void SetMainRenderTarget(const CoreGraphics::RenderTarget &amp;rt)</strong></td>
<td>set the main render target</td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; GetMainRenderTarget() const</strong></td>
<td>get the main render target</td>
</tr>
<tr>
<td><strong>void AddRenderTarget(const Resources::ResourceId &amp;resId, const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp;rt)</strong></td>
<td>add a render target to the frame</td>
</tr>
<tr>
<td><strong>SizeT GetNumRenderTargets() const</strong></td>
<td>get number of render targets</td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; GetRenderTargetByIndex(IndexT i)</strong></td>
<td>get render target by index</td>
</tr>
<tr>
<td><strong>bool HasRenderTarget(const Resources::ResourceId &amp;resId)</strong></td>
<td>return true if render target exists by name</td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; GetRenderTargetByName(const Resources::ResourceId &amp;resId)</strong></td>
<td>get render target by name</td>
</tr>
<tr>
<td>Type</td>
<td>Method Name</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>void</td>
<td>AddMultipleRenderTarget</td>
</tr>
<tr>
<td>SizeT</td>
<td>GetNumMultipleRenderTarget</td>
</tr>
<tr>
<td>constPtr<a href="">CoreGraphics::MultipleRenderTarget</a> &amp;</td>
<td>GetMultipleRenderTargetByIndex</td>
</tr>
<tr>
<td>bool</td>
<td>HasMultipleRenderTarget</td>
</tr>
<tr>
<td>constPtr<a href="">CoreGraphics::MultipleRenderTarget</a> &amp;</td>
<td>GetMultipleRenderTargetByName</td>
</tr>
<tr>
<td>void</td>
<td>AddTexture</td>
</tr>
<tr>
<td>SizeT</td>
<td>GetNumTextures</td>
</tr>
<tr>
<td>constPtr/Resources::ManagedTexture&gt; &amp;</td>
<td>GetTextureByIndex</td>
</tr>
<tr>
<td>bool</td>
<td>HasTexture</td>
</tr>
<tr>
<td>constPtr/Resources::ManagedTexture&gt; &amp;</td>
<td>GetTextureByName</td>
</tr>
<tr>
<td>void</td>
<td>AddFramePassBase</td>
</tr>
<tr>
<td>Function/Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>GetNumFramePassBases() const</td>
<td>get number of frame passes</td>
</tr>
<tr>
<td>GetFramePassBaseByIndex(i) const</td>
<td>get frame pass by index</td>
</tr>
<tr>
<td>HasFramePassBase(const Resources::ResourceId &amp;resId) const</td>
<td>return true if names pass exists</td>
</tr>
<tr>
<td>GetFramePassBaseByIndex(const Resources::ResourceId &amp;resId) const</td>
<td>get frame pass by name</td>
</tr>
<tr>
<td>GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>IsInstanceOf(const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>IsInstanceOf(const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>IsA(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given a derived class</td>
</tr>
<tr>
<td>IsA(const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given a derived class by string</td>
</tr>
<tr>
<td>IsA(const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String &amp;</code></td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td><code>Util::FourCC</code></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void DumpRefCountingLeaks()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Frame::FrameShaderLoader
Frame::FrameShaderLoader Class Reference

#include <frameshaderloader.h>
Detailed Description

**Loader** class to load frame shader from XML stream.

(C) 2007 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Ptr&lt; FrameShader &gt; LoadFrameShader</code> (const <code>Resources::ResourceId</code> &amp;name, const <code>IO::URI</code> &amp;uri)</td>
<td>load a frame shader from an XML file</td>
</tr>
</tbody>
</table>
Frame::LightingMode
Frame::LightingMode Class Reference

#include <lightingmode.h>
Detailed Description

The lighting mode to perform when rendering a frame batch.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lighting mode enum</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FromString</strong></td>
<td>convert from string</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>convert to string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](http://doxygen.net) at Fri Mar 26 15:21:44 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Frame::SortingMode
Frame::SortingMode Class Reference

#include <sortingmode.h>
Detailed Description

The sorting mode to perform when rendering a frame batch.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>Code</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>sorting mode enum</em></td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static Code</th>
<th>FromString (const Util::String &amp;str)</th>
</tr>
</thead>
<tbody>
<tr>
<td>convert from string</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static Util::String</th>
<th>ToString (Code c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>convert to string</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:44 2010
FrameSync::FrameSyncHandlerThread
FrameSync::FrameSyncHandlerThread
Class Reference

#include <framesynchandlethreads.h>

Inheritance diagram for FrameSync::FrameSyncHandlerThread:
Detailed Description

A special handler thread class for synchronizing the render thread and the game thread (and possibly other threads which need to run in lock-step with the render thread).

The FrameSyncHandlerThread object implements a clearly defined sync point which separates the previous frame from the next frame for all participating threads:

- all threads arrive at the frame sync point and wait for the arrival of the other threads
- once all threads have arrived:
  - advance the master time
  - flip message queues
  - start all threads

(C) 2009 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>Priority</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>thread priorities</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>FrameSyncHandlerThread ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~FrameSyncHandlerThread ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>void EnterLockStepMode ()</code></td>
<td>enter lock-step mode</td>
</tr>
<tr>
<td><code>void LeaveLockStepMode ()</code></td>
<td>leave lock-step mode</td>
</tr>
<tr>
<td><code>bool LockStepModeActive () const</code></td>
<td>return true if currently in lock-step mode</td>
</tr>
<tr>
<td><code>IndexT GetFrameCount () const</code></td>
<td>get the current frame count (also available on <code>FrameSyncTimer</code>)</td>
</tr>
<tr>
<td><code>void StartFixedFrameTime (Timing::Time frameTime)</code></td>
<td>start fixed frame time mode</td>
</tr>
<tr>
<td><code>void StopFixedFrameTime ()</code></td>
<td>stop fixed frame time mode</td>
</tr>
<tr>
<td><code>bool IsFixedFrameTime ()</code></td>
<td>check whether the fixed frame time mode is currently active</td>
</tr>
<tr>
<td><code>void ArriveAtSyncPoint (bool masterThread)</code></td>
<td>notify arrival at sync point</td>
</tr>
<tr>
<td><code>virtual void AddMessage (const Ptr&lt; Messaging::Message &gt; &amp;msg)</code></td>
<td>add a message to be handled</td>
</tr>
<tr>
<td><code>virtual void CancelMessage (const Ptr&lt; Messaging::Message &gt; &amp;msg)</code></td>
<td>cancel a pending message</td>
</tr>
<tr>
<td><code>virtual void WaitForMessage (const Ptr&lt; Messaging::Message &gt; &amp;msg)</code></td>
<td>wait for message to be handled</td>
</tr>
<tr>
<td><code>virtual void DoWork ()</code></td>
<td>this method runs in the thread context</td>
</tr>
<tr>
<td><code>void AttachHandler (const Ptr&lt; Handler &gt; &amp;h)</code></td>
<td>attach a message handler</td>
</tr>
</tbody>
</table>
void **RemoveHandler** (const **Ptr**< **Handler** > &h)
"dynamically remove a message handler"

void **ClearHandlers** ()
"clear all attached message handlers"

void **WaitForHandlersOpened** ()
"wait until handlers have been opened"

void **SetPriority** (**Priority** p)
"set the thread priority"

**Priority** **GetPriority** () const
"get the thread priority"

void **SetCoreId** (System::Cpu::CoreId coreId)
"set cpu core on which the thread should be running"

System::Cpu::CoreId **GetCoreId** () const
"get the cpu core on which the thread should be running"

void **SetStackSize** (**SizeT** s)
"set stack size in bytes (default is 4 KByte)"

**SizeT** **GetStackSize** () const
"get stack size"

void **SetName** (const **Util::String** &n)
"set thread name"

const **Util::String** & **GetName** () const
"get thread name"

void **Start** ()
"start executing the thread code, returns when thread has actually started"

void **Stop** ()
"request threading code to stop, returns when thread has actually finished"

bool **IsRunning** () const
"return true if thread has been started"

int **GetRefCount** () const
"get the current refcount"

void **AddRef** ()
"increment refcount by one"

void **Release** ()
"decrement refcount and destroy object if refcount is zero"

bool **IsInstanceOf** (const **Rtti** &rtti) const
```
bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
```
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static void</strong> YieldThread ()</td>
<td>yield the thread (gives up current time slice)</td>
</tr>
<tr>
<td><strong>static void</strong> SetMyThreadName(const char *n)</td>
<td>set thread name from within thread context</td>
</tr>
<tr>
<td>**static const char ** GetMyThreadName ()</td>
<td>obtain name of thread from within thread context</td>
</tr>
<tr>
<td><strong>static Threading::ThreadId</strong> GetMyThreadId ()</td>
<td>get the thread ID of this thread</td>
</tr>
<tr>
<td><strong>static void</strong> DumpRefCountingLeaks ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ThreadOpenHandlers()</code></td>
<td>Open message handlers</td>
</tr>
<tr>
<td><code>ThreadCloseHandlers()</code></td>
<td>Close message handlers</td>
</tr>
<tr>
<td><code>ThreadUpdateHandlers()</code></td>
<td>Open dynamically added handlers, and call <code>DoWork()</code> on all attached handlers</td>
</tr>
<tr>
<td><code>ThreadUpdateDeferredMessages()</code></td>
<td>Update deferred messages, return true if at least one message has been handled</td>
</tr>
<tr>
<td><code>ThreadDiscardDeferredMessages()</code></td>
<td>Clear leftover deferred messages</td>
</tr>
<tr>
<td><code>ThreadHandleMessages(const Util::Array&lt;Ptr&lt;Message&gt; &gt; &amp;msgArray)</code></td>
<td>Handle messages in array, return true if at least one message has been handled</td>
</tr>
<tr>
<td><code>ThreadHandleSingleMessage(const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>Handle a single message without deferred support, return if message has been handled</td>
</tr>
<tr>
<td><code>ThreadSignalMessageHandled()</code></td>
<td>Signal message handled event (only call if at least one message has been handled)</td>
</tr>
<tr>
<td><code>EmitWakeupSignal()</code></td>
<td>Override this method if your thread loop needs a wakeup call before stopping</td>
</tr>
<tr>
<td><code>ThreadStopRequested()</code></td>
<td>Check if stop is requested, call from <code>DoWork()</code> to see if the thread proc should quit</td>
</tr>
</tbody>
</table>
Member Function Documentation

void FrameSync::FrameSyncHandlerThread::EnterLockStepMode()

enter lock-step mode

Enter lock-step mode, this method must only be called from the slave thread.

void FrameSync::FrameSyncHandlerThread::LeaveLockStepMode()

leave lock-step mode

Leave lock-step mode, this method must only be called from the slave thread.

void FrameSync::FrameSyncHandlerThread::StartFixedFrameTime(Timing::Time frameTime)

start fixed frame time mode

Start the fixed frame time mode. In this mode the current time is increased by a fixed value every frame.

void FrameSync::FrameSyncHandlerThread::StopFixedFrameTime()

stop fixed frame time mode

Stop the fixed frame time mode.

void FrameSync::FrameSyncHandlerThread::ArriveAtSyncPoint(bool masterThread)

notify arrival at sync point

This is the central sync point. Every thread will call this method at the start of a new frame to synchronize with the other lock-step frames.
Once all threads have arrived, the master time will be advanced and the double-buffered message queues will be flipped.

```cpp
void FrameSync::FrameSyncHandlerThread::AddMessage(const Ptr<Messaging::Message> &msg) [virtual]
```

add a message to be handled

Add a message to the producer queue, since we're using double buffering so we don't need thread-synchronization.

Reimplemented from `Messaging::HandlerThreadBase`.

```cpp
void FrameSync::FrameSyncHandlerThread::CancelMessage(const Ptr<Messaging::Message> &msg) [virtual]
```

cancel a pending message

Cancel a pending message from the producer queue. If the message is already in the consumer queue, then it's too late to cancel the message!

Reimplemented from `Messaging::HandlerThreadBase`.

```cpp
void FrameSync::FrameSyncHandlerThread::WaitForMessage(const Ptr<Messaging::Message> &msg) [virtual]
```

wait for message to be handled

Wait for a message to be handled.

Reimplemented from `Messaging::HandlerThreadBase`.

```cpp
void FrameSync::FrameSyncHandlerThread::DoWork() [virtual]
```

this method runs in the thread context

The central message processing loop.
Reimplemented from `Win360::Win360Thread`.

```cpp
void Messaging::HandlerThreadBase::AttachHandler(const Ptr<Handler> & h) [inherited]
```

attach a message handler

Attach a message handler to the port. This method may be called from any thread.

```cpp
void Messaging::HandlerThreadBase::RemoveHandler(const Ptr<Handler> & h) [inherited]
```

dynamically remove a message handler

Remove a message handler. This method may be called from any thread.

```cpp
void Messaging::HandlerThreadBase::ClearHandlers() [inherited]
```

clear all attached message handlers

This clears all attached message handlers.

```cpp
void Messaging::HandlerThreadBase::WaitForHandlersOpened() [inherited]
```

wait until handlers have been opened

Wait on the handlers-opened event (will be signalled by the ThreadOpenHandlers method.

```cpp
void Messaging::HandlerThreadBase::ThreadOpenHandlers() [protected, inherited]
```

open message handlers

Open attached message handlers. This method must be called at the
start of the handler thread.

```cpp
void Messaging::HandlerThreadBase::ThreadCloseHandlers() [protected, inherited]
```

close message handlers

Close attached message handlers. This method must be called right before the handler thread shuts down.

```cpp
void Messaging::HandlerThreadBase::ThreadUpdateHandlers() [protected, inherited]
```

open dynamically added handlers, and call `DoWork()` on all attached handlers

Do per-frame update of attached handlers. This will open handlers which have been added late, and call the `DoWork()` method on handlers from within the thread context.

```cpp
bool Messaging::HandlerThreadBase::ThreadUpdateDeferredMessages() [protected, inherited]
```

update deferred messages, return true if at least one message has been handled

This checks every message in the deferred message array whether it has been handled yet, if yes, the message's actual handled flag will be set, and the message will be removed from the deferred handled array. If at least one message has been handled, the method will return true, if no message has been handled, the method returns false. If message have been handled, don't forget to call `ThreadSignalMessageHandled()` later!

```cpp
void Messaging::HandlerThreadBase::ThreadDiscardDeferredMessages() [protected, inherited]
```

clear leftover deferred messages

This clears any leftover deferred messages. Call right before shutdown of the handler thread.
handle messages in array, return true if at least one message has been handled

Handle all message in the provided message array. Supports batched and deferred messages. Calls `ThreadHandleSingleMessage()`. If at least one message has been handled, the method returns true.

handle a single message without deferred support, return if message has been handled

Handle a single message, called by `ThreadHandleMessages()`. Return true if message has been handled. This method MUST be called from `ThreadHandleMessages()`, since this method will not explicitely take the handlers array critical section.

signal message handled event (only call if at least one message has been handled)

Signal the message-handled flag. Call this method once per handler-loop if either `ThreadUpdateDeferredMessages` or `ThreadHandleMessages` returns true!

set thread name

Set the thread's name. To obtain the current thread's name from
anywhere in the thread's execution context, call the static method
Thread::GetMyThreadName().

const Util::String &
Win360::Win360Thread::GetName () const [inline, inherited]

get thread name

Get the thread's name. This is the vanilla method which returns the name member. To obtain the current thread's name from anywhere in the thread's execution context, call the static method Thread::GetMyThreadName().

void
Win360::Win360Thread::Start () [inherited]

start executing the thread code, returns when thread has actually started

Start the thread, this creates a Win32 thread and calls the static ThreadProc, which in turn calls the virtual DoWork() class of this object. The method waits for the thread to start and then returns.

void
Win360::Win360Thread::Stop () [inherited]

request threading code to stop, returns when thread has actually finished

This stops the thread by signalling the stopRequestEvent and waits for the thread to actually quit. If the thread code runs in a loop it should use the IsStopRequested() method to see if the thread object wants it to shutdown. If so DoWork() should simply return.

Reimplemented in Jobs::TPWorkerThread.

bool
Win360::Win360Thread::IsRunning () const [inherited]

return true if thread has been started
Returns true if the thread is currently running.

```cpp
void Win360::Win360Thread::YieldThread() [static, inherited]
```

yield the thread (gives up current time slice)

The yield function is empty on **Win32** and Xbox360.

```cpp
void Win360::Win360Thread::SetMyThreadName(const char * n) [static, inherited]
```

set thread name from within thread context

Static method which sets the name of this thread. This is called from within ThreadProc. The string pointed to must remain valid until the thread is terminated!

```cpp
const char * Win360::Win360Thread::GetMyThreadName() [static, inherited]
```

obtain name of thread from within thread context

Static method to obtain the current thread name from anywhere in the thread's code.

```cpp
Threading::ThreadId Win360::Win360Thread::GetMyThreadId() [static, inherited]
```

get the thread ID of this thread

Static method which returns the ThreadId of this thread.

```cpp
void Win360::Win360Thread::EmitWakeupSignal() [protected, virtual, inherited]
```

override this method if your thread loop needs a wakeup call before stopping

This method is called by **Thread::Stop()** after setting the stopRequest event and before waiting for the thread to stop. If your thread runs a
loop and waits for jobs it may need an extra wakeup signal to stop waiting and check for the \texttt{ThreadStopRequested()} event. In this case, override this method and signal your event object.

Reimplemented in \texttt{Jobs::TPWorkerThread}, and \texttt{Messaging::BlockingHandlerThread}.

```cpp
bool
Win360::Win360Thread::ThreadStopRequested ( ) const [inline, protected, inherited]
```

check if stop is requested, call from \texttt{DoWork()} to see if the thread proc should quit

If the derived \texttt{DoWork()} method is running in a loop it must regularly check if the process wants the thread to terminate by calling \texttt{ThreadStopRequested()} and simply return if the result is true. This will cause the thread to shut down.

```cpp
int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void
Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const \texttt{Util::String} &
Core::RefCounted::GetClassName ( ) const [inline, inherited]
```
get the class name

Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
FrameSync::FrameSyncSharedData
FrameSync::FrameSyncSharedData
Class Reference

#include <framesyncshareddata.h>

Inheritance diagram for FrameSync::FrameSyncSharedData:
Detailed Description

A lock-less, double buffered data exchange object between 2 threads which are running in lockstep through the FrameSync system (e.g. the main thread and render thread). The owner thread is defined as the thread which owns the FrameSyncSharedData object (usually the main thread), and the client thread is usually the render thread.

(C) 2010 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FrameSyncSharedData()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~FrameSyncSharedData()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>*<em>template&lt;typename T&gt; void OwnerSetup (T <em>ignoreMe=0)</em></em></td>
<td>Owner-side setup function</td>
</tr>
<tr>
<td>*<em>template&lt;typename T&gt; void OwnerDiscard (T <em>ignoreMe=0)</em></em></td>
<td>Owner-discard, must be called when owner is done with the object</td>
</tr>
<tr>
<td><strong>template&lt;typename T&gt; T &amp; Owner ()</strong></td>
<td>Access data as owner</td>
</tr>
<tr>
<td>*<em>template&lt;typename T&gt; void ClientSetup (T <em>ignoreMe=0)</em></em></td>
<td>Client-side setup function</td>
</tr>
<tr>
<td>*<em>template&lt;typename T&gt; void ClientDiscard (T <em>ignoreMe=0)</em></em></td>
<td>Client-discard, must be called when the client is done with the object</td>
</tr>
<tr>
<td><strong>template&lt;typename T&gt; T &amp; Client ()</strong></td>
<td>Access data as client</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

template<typename TYPE>
void
FrameSync::FrameSyncSharedData::OwnerSetup ( TYPE ignoreMe = 0 ) [inline]

owner-side setup function

Call this method from the owner thread before handing a smart ptr to the object to the client thread!

template<typename TYPE>
void
FrameSync::FrameSyncSharedData::OwnerDiscard ( TYPE ignoreMe = 0 ) [inline]

owner-discard, must be called when owner is done with the object

Call this method from the owner thread when the owner thread no longer needs the object.

template<typename TYPE>
void
FrameSync::FrameSyncSharedData::ClientSetup ( TYPE ignoreMe = 0 ) [inline]

client-side setup function

Call this method on the client thread. It must be guaranteed that OwnerSetup() has already finished when this method is called! The provided template type must be identical between all method calls!

template<typename TYPE>
void
FrameSync::FrameSyncSharedData::ClientDiscard ( TYPE ignoreMe = 0 ) [inline]

client-discard, must be called when the client is done with the object

Call this method from the client thread when the client thread no longer needs the object.

int const [inline, inherited]
Core::RefCounted::GetRefCount ( )

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:45 2010
FrameSync::FrameSyncTimer
FrameSync::FrameSyncTimer Class Reference

#include <framesynctimer.h>

Inheritance diagram for FrameSync::FrameSyncTimer:
Detailed Description

A thread-local time source object which is synchronized with the sync point in the FrameSyncHandlerThread. Time values are thread-locally cached and thus no thread-synchronization is necessary when reading time values. FrameSyncTimer objects are updated inside the FrameSyncHandler::ArriveAtSyncPoint() method.

Threads interested in the master time create a FrameSyncTimer singleton and register it with FrameSyncHandlerThread through the GraphicsInterface singleton.

Please note that when calling the Start()/Stop()/Reset() methods, that the ENTIRE master time will be affected (these methods go straight through to the FrameSyncHandler object).

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FrameSyncTimer ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~FrameSyncTimer ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void Setup ()</td>
<td>setup the object</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>discard the object</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>return true if object is valid</td>
</tr>
<tr>
<td>void UpdateTimePolling ()</td>
<td>update the time through polling, only necessary for threads other than render/game thread!</td>
</tr>
<tr>
<td>void StartTime ()</td>
<td>start the master time (DON'T CALL FREQUENTLY!)</td>
</tr>
<tr>
<td>void StopTime ()</td>
<td>stop the master time (DON'T CALL FREQUENTLY!)</td>
</tr>
<tr>
<td>void ResetTime ()</td>
<td>reset the master time (DON'T CALL FREQUENTLY!)</td>
</tr>
<tr>
<td>bool IsTimeRunning () const</td>
<td>return true if master time is running (DON'T CALL FREQUENTLY!)</td>
</tr>
<tr>
<td>Timing::Time GetTime () const</td>
<td>get current time</td>
</tr>
<tr>
<td>Timing::Tick GetTicks () const</td>
<td>get current time in ticks</td>
</tr>
<tr>
<td>Timing::Time GetFrameTime () const</td>
<td>get current frame time</td>
</tr>
<tr>
<td>Timing::Tick GetFrameTicks () const</td>
<td>get current frame time in ticks</td>
</tr>
<tr>
<td>IndexT GetFrameCount () const</td>
<td>get current frame count</td>
</tr>
<tr>
<td>void SetTimeFactor (Timing::Time factor)</td>
<td>set time factor</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetTimeFactor () const</code></td>
<td>get time factor</td>
</tr>
<tr>
<td><code>GetScaledTime () const</code></td>
<td>get scaled time</td>
</tr>
<tr>
<td><code>GetScaledFrameTime () const</code></td>
<td>get scaled time</td>
</tr>
<tr>
<td><code>GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void FrameSync::FrameSyncTimer::Setup() {
    setup the object

    Query master time from the FrameSyncHandlerThread once at creation time. This is safe but slow, but necessary to immediately provide a valid time stamp in the first frame.

    void FrameSync::FrameSyncTimer::UpdateTimePolling() {
        update the time through polling, only necessary for threads other then render/game thread!

        Update the time by polling the FrameSyncHandlerThread singleton. This requires a CriticalSection to be taken and should not be used for threads which run in lockstep mode (i.e. the game thread or the render thread) since these will update their time automatically during the frame-sync-point.

    } int Core::RefCounted::GetRefCount() const [inline, inherited] {
        get the current refcount

        Return the current refcount of the object.

    } void Core::RefCounted::AddRef() [inline, inherited] {
        increment refcount by one

        Increment the refcount of the object.

    } void Core::RefCounted::Release() [inline, inherited] {
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
FSM::State
FSM::State Class Reference

#include <state.h>

Inheritance diagram for FSM::State:
Detailed Description

Implements a state of a state machine.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~<strong>State</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>SetName</strong> (const <strong>Util::String</strong> &amp;n)</td>
<td>set name of the state</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetName</strong> () const</td>
</tr>
<tr>
<td>void <strong>AddEntryAction</strong> (const <strong>Ptr<a href="">Actions::ActionList</a></strong> &amp;action)</td>
<td>add a state entry action</td>
</tr>
<tr>
<td>void <strong>AddFrameAction</strong> (const <strong>Ptr<a href="">Actions::ActionList</a></strong> &amp;action)</td>
<td>add a state frame action</td>
</tr>
<tr>
<td>void <strong>AddExitAction</strong> (const <strong>Ptr<a href="">Actions::ActionList</a></strong> &amp;action)</td>
<td>add a state exit action</td>
</tr>
<tr>
<td>void <strong>AddTransition</strong> (const <strong>Ptr&lt;Transition&gt;</strong> &amp;transition)</td>
<td>add a transition object to the state</td>
</tr>
<tr>
<td>void <strong>OnEntry</strong> ()</td>
<td>called when state is entered</td>
</tr>
<tr>
<td>void <strong>OnFrame</strong> ()</td>
<td>called per-frame while state is active</td>
</tr>
<tr>
<td>void <strong>OnExit</strong> ()</td>
<td>called when state is left</td>
</tr>
<tr>
<td>void <strong>Notify</strong> (const <strong>Ptr<a href="">Messaging::Message</a></strong> &amp;msg)</td>
<td>notify the state about incoming message</td>
</tr>
<tr>
<td><strong>Ptr&lt;Transition&gt;</strong></td>
<td><strong>EvaluateTransitions</strong> ()</td>
</tr>
<tr>
<td>int <strong>GetNumTransitions</strong> ()</td>
<td>get number of transitions</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>const Ptr&lt; Transition &gt;&amp;</code> GetTransitionAtIndex(int index) const</td>
<td>get pointer to transition at index</td>
</tr>
<tr>
<td><code>const Ptr&lt; Actions::SequenceAction &gt;&amp;</code> GetFrameActions() const</td>
<td>get all frame actions</td>
</tr>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA(const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA(const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName() const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC() const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void FSM::State::OnEntry()
```
called when state is entered
Executes the entry actions. This method is called once by the state machine when the state becomes the active state.

```cpp
void FSM::State::OnFrame()
```
called per-frame while state is active
Executes the frame actions. This method is called once per frame by the state machine while this state is the active state.

```cpp
void FSM::State::OnExit()
```
called when state is left
Executes the exit actions. This method is called once per frame by the state machine before this state becomes inactive.

```cpp
void FSM::State::Notify(const Ptr<Messaging::Message> msg &)
```
notify the state about incoming message
Notifies the state about any incoming messages. The method call is simply forwarded to all FrameActions and Transitions. Usually, only FSMActions and FSMConditions are interested about messages.

```cpp
Ptr<Transition>
FSM::State::EvaluateTransitions()
```
evaluate transitions, return first transition which evaluates true
This evaluates all transitions. The first which evaluates to true will be returned, or 0 if no transition evaluates to true. The caller must care about executing the transition actions and switching states.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

FSM::StateMachine
#include <statemachine.h>

Inheritance diagram for FSM::StateMachine:
Detailed Description

Implements the state machine for the FSM subsystem.

Note: On first start the state "Start" will be started if available, otherwise the first in row.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StateMachine ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~StateMachine ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void SetName (const Util::String &amp;n)</code></td>
<td>set the name of the state machine</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetName () const</code></td>
<td>get name of state machine</td>
</tr>
<tr>
<td><code>void SetEntity (const Ptr&lt;Game::Entity&gt; &amp;e)</code></td>
<td>set pointer to entity which owns the state machine</td>
</tr>
<tr>
<td><code>constPtr&lt;Game::Entity&gt; &amp; GetEntity () const</code></td>
<td>get pointer to entity</td>
</tr>
<tr>
<td><code>void AddState (const Ptr&lt;State&gt; &amp;s)</code></td>
<td>add a state to the state machine</td>
</tr>
<tr>
<td><code>Ptr&lt;State&gt; FindStateByName (const Util::String &amp;n)</code></td>
<td>find a state by name</td>
</tr>
<tr>
<td><code>void Open ()</code></td>
<td>called when the state machine is initially opened</td>
</tr>
<tr>
<td><code>void Close ()</code></td>
<td>called when the state machine is closed</td>
</tr>
<tr>
<td><code>bool IsOpen () const</code></td>
<td>return true if currently open</td>
</tr>
<tr>
<td><code>bool Update ()</code></td>
<td>update the state machine while open (called per-frame)</td>
</tr>
<tr>
<td><code>constPtr&lt;State&gt; &amp; GetCurrentState () const</code></td>
<td>get current state</td>
</tr>
<tr>
<td><code>void Notify (const Ptr&lt;Message::Message&gt; &amp;msg)</code></td>
<td>notify state machine about a message</td>
</tr>
<tr>
<td><code>Util::String GetDebugTxt () const</code></td>
<td>get current debug txt</td>
</tr>
<tr>
<td><code>Math::float4 GetDebugColor () const</code></td>
<td>get debug color</td>
</tr>
<tr>
<td>int</td>
<td>GetRefCount () const</td>
</tr>
<tr>
<td>------</td>
<td>----------------------</td>
</tr>
<tr>
<td></td>
<td>get the current refcount</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>AddRef ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>increment refcount by one</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>Release ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::String &amp;className) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
add a state to the state machine

Add a state to the state machine.

find a state by name

Find a state by its name. Returns 0 if not found.

called when the state machine is initially opened

Open the state machine. This will set the first state in the state array as active state.

called when the state machine is closed

Close the state machine. This will unset the current state, so that it may cleanup itself properly.

update the state machine while open (called per-frame)

This updates the state machine, must be called frequently while the
state machine is open.

const Ptr< State > &
FSM::StateMachine::GetCurrentState( ) const

get current state

Returns the current state.

void
FSM::StateMachine::Notify< ( Messaging::Message msg ) > &
notify state machine about a message

This notifies the current state about a message.

int
Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name
Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

FSM::Transition
#include <transition.h>

Inheritance diagram for FSM::Transition:
Detailed Description

Implements a state transition. If no target state name if given, the previously active state will be activated when the transition conditions are met.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Transition ()</code></td>
<td>virtual constructor</td>
</tr>
<tr>
<td><code>~Transition ()</code></td>
<td>virtual destructor</td>
</tr>
<tr>
<td><code>OnActivate ()</code></td>
<td>void called when parent states becomes active</td>
</tr>
<tr>
<td><code>OnDeactivate ()</code></td>
<td>void called when parent states becomes inactive</td>
</tr>
<tr>
<td><code>Notify (const Ptr&lt; Messaging::Message &gt; &amp;msg)</code></td>
<td>void notify FSMConditions about message</td>
</tr>
<tr>
<td><code>SetTargetState (const Util::String &amp;n)</code></td>
<td>void set name of target state</td>
</tr>
<tr>
<td><code>GetTargetState () const</code></td>
<td>const get name of target state</td>
</tr>
<tr>
<td><code>AddCondition (const Ptr&lt; Conditions::Condition &gt; &amp;condition)</code></td>
<td>void add a condition to the transition's AND condition block</td>
</tr>
<tr>
<td><code>AddAction (const Ptr&lt; Actions::ActionList &gt; &amp;action)</code></td>
<td>void add a transition action</td>
</tr>
<tr>
<td><code>EvaluateConditions ()</code></td>
<td>bool evaluate transition conditions</td>
</tr>
<tr>
<td><code>ExecuteActions ()</code></td>
<td>void execute actions associated with transition</td>
</tr>
<tr>
<td><code>GetRefCount () const</code></td>
<td>int get the current refcount</td>
</tr>
<tr>
<td><code>AddRef ()</code></td>
<td>void increment refcount by one</td>
</tr>
<tr>
<td><code>Release ()</code></td>
<td>void decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>bool return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::String &amp;className)</code></td>
<td>bool return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td>const</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>return true if this object is instance of given class by string</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class by fourcc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class, or a derived class</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;  </th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class name</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC  </th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</code></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void FSM::Transition::OnActivate()
```
called when parent states becomes active
This method gets called when the parent state becomes active.

```cpp
void FSM::Transition::OnDeactivate()
```
called when parent states becomes inactive
This method gets called when the parent state becomes inactive.

```cpp
void FSM::Transition::Notify(const Ptr<Messaging::Message> &msg)
```
notify FSMConditions about message
This method notifies FSMConditions about any incoming messages they are interested in.

```cpp
bool FSM::Transition::EvaluateConditions()
```
evaluate transition conditions
This evaluates the state of all conditions and ANDs them together. If there are no Conditions, the transition is be true.

```cpp
void FSM::Transition::ExecuteActions()
```
execute actions associated with transition
This executes all actions added to the transition.

int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Game::BaseGameFeatureUnit
Game::BaseGameFeatureUnit Class Reference

#include <basegamefeatureunit.h>
Detailed Description

The **BaseGameFeatureUnit** creates everything to allow load and run a game level. Therefore it creates managers to allow creation and handling of entities and its properties. It supports loading of a game level from a database with the db server and offers simple methods for opening a new game, loading a level or a savegame. The Feature also creates the timemanager and different timesources used by subsystems.

If you wanna use your own managers (other specialized entitymanager), derive from this class and overwrite OnActivate() OnDeactivate().

(C) 2007 Radon Labs GmbH
Game::Entity
Game::Entity Class Reference

#include <entity.h>

Inheritance diagram for Game::Entity:
Detailed Description

The game entity class. A game entity represents one game object, such as an actor, an item, etc...

The entity class itself doesn't have any game specific attributes or functionality. Instead it's configured by attaching properties to the entity. Properties add functionality to a game entity, for instance, if an entity should be visible, add a GraphicsProperty, if it should be audible, add an AudioProperty, and so forth...

Any data used for initializing any property or holding any information about states etc. is stored in the attributeTable.

(C) 2007 RadonLabs GmbH
Public Types

typedef unsigned int EntityId

an Id type, used to identify entities
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~Entity ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetCategory () const</strong></td>
<td>Get the entity's category</td>
</tr>
<tr>
<td><strong>const EntityId GetUniqueld () const</strong></td>
<td>Get unique id of entity</td>
</tr>
<tr>
<td><strong>bool AcceptsMessage (const Messaging::Id &amp;msgId) const</strong></td>
<td>Return true if any property accepts/processes message</td>
</tr>
<tr>
<td><strong>void SendSync (const Ptr<a href="">Messaging::Message</a> &amp;msg)</strong></td>
<td>Send a synchronous message to the entity</td>
</tr>
<tr>
<td><strong>bool IsActive () const</strong></td>
<td>Return true if the entity is currently active (between OnActivate/OnDeactivate)</td>
</tr>
<tr>
<td><strong>const Ptr<a href="">Db::ValueTable</a> &amp; GetAttrTable () const</strong></td>
<td>Get the instance attribute table for the entity</td>
</tr>
<tr>
<td><strong>IndexT GetAttrTableRowIndex () const</strong></td>
<td>Get the instance attribute table row index for the entity</td>
</tr>
<tr>
<td><strong>Ptr&lt; Property &gt; FindProperty (const Core::Rtti &amp;rtti) const</strong></td>
<td>Find property by RTTI (performs Isa() check), returns invalid pointer if not exists, slow!</td>
</tr>
<tr>
<td><strong>void RegisterPropertyCallback (const Ptr&lt;Property&gt; &amp;prop, Property::CallbackType callback)</strong></td>
<td>Register a property callback, called by Property::SetupCallback() method</td>
</tr>
<tr>
<td><strong>void OnActivate ()</strong></td>
<td>Called when attached to world</td>
</tr>
<tr>
<td><strong>void OnDeactivate ()</strong></td>
<td>Called when removed from world</td>
</tr>
<tr>
<td><strong>void OnBeginFrame ()</strong></td>
<td></td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>OnMoveBefore()</code></td>
<td>called at the beginning of the frame before movement</td>
</tr>
<tr>
<td><code>OnMoveAfter()</code></td>
<td>called after movement</td>
</tr>
<tr>
<td><code>OnRender()</code></td>
<td>called before rendering</td>
</tr>
<tr>
<td><code>OnRenderDebug()</code></td>
<td>debug rendering called before rendering</td>
</tr>
<tr>
<td><code>OnLoseActivity()</code></td>
<td>called if entity loses activity</td>
</tr>
<tr>
<td><code>OnGainActivity()</code></td>
<td>called if entity gains activity</td>
</tr>
<tr>
<td><code>OnLoad()</code></td>
<td>called after loading from database has happened</td>
</tr>
<tr>
<td><code>OnStart()</code></td>
<td>called when the entity starts to live in the complete world</td>
</tr>
<tr>
<td><code>OnSave()</code></td>
<td>called before saving from database has happened</td>
</tr>
<tr>
<td><code>HasAttr(const Attr::AttrId &amp;attrId) const</code></td>
<td>return true if entity has an attribute</td>
</tr>
<tr>
<td><code>SetAttr(const Attr::Attribute &amp;attr)</code></td>
<td>generic attribut setter (slow!)</td>
</tr>
<tr>
<td><code>GetAttr(const Attr::AttrId &amp;attrId) const</code></td>
<td>generic attribute getter (slow!)</td>
</tr>
<tr>
<td><code>SetAttrValue(const Attr::AttrId &amp;attrId, const Util::Variant &amp;val)</code></td>
<td>generic attribut set with variant</td>
</tr>
<tr>
<td><code>SetString(const Attr::StringAttrId &amp;attrId, const Util::String &amp;s)</code></td>
<td>set string attribute on the entity</td>
</tr>
<tr>
<td><code>GetString(const Attr::StringAttrId &amp;attrId)</code></td>
<td>get string attribute from the entity</td>
</tr>
<tr>
<td><code>SetInt(const Attr::IntAttrId &amp;attrId, int i)</code></td>
<td>set int attribute on the entity</td>
</tr>
<tr>
<td><code>int</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>GetInt</code></td>
<td>Const <code>Attr::IntAttrId &amp;attrId)</code> const</td>
</tr>
<tr>
<td><code>SetFloat</code></td>
<td>(Const Attr::FloatAttrId &amp;attrId, float f)</td>
</tr>
<tr>
<td><code>GetFloat</code></td>
<td>(Const Attr::FloatAttrId &amp;attrId) const</td>
</tr>
<tr>
<td><code>SetBool</code></td>
<td>(Const Attr::BoolAttrId &amp;attrId, bool b)</td>
</tr>
<tr>
<td><code>GetBool</code></td>
<td>(Const Attr::BoolAttrId &amp;attrId) const</td>
</tr>
<tr>
<td><code>SetFloat4</code></td>
<td>(Const Attr::Float4AttrId &amp;attrId, const <code>Math::float4</code> &amp;v)</td>
</tr>
<tr>
<td><code>GetFloat4</code></td>
<td>(Const Attr::Float4AttrId &amp;attrId) const</td>
</tr>
<tr>
<td><code>SetMatrix44</code></td>
<td>(Const Attr::Matrix44AttrId &amp;attrId, const <code>Math::matrix44</code> &amp;m)</td>
</tr>
<tr>
<td><code>GetMatrix44</code></td>
<td>(Const Attr::Matrix44AttrId &amp;attrId) const</td>
</tr>
<tr>
<td><code>SetGuid</code></td>
<td>(Const Attr::GuidAttrId &amp;attrId, const <code>Util::Guid</code> &amp;guid)</td>
</tr>
<tr>
<td><code>GetGuid</code></td>
<td>(Const Attr::GuidAttrId &amp;attrId) const</td>
</tr>
<tr>
<td><code>SetBlob</code></td>
<td>(Const Attr::BlobAttrId &amp;attrId, const <code>Util::Blob</code> &amp;blob)</td>
</tr>
<tr>
<td><code>GetBlob</code></td>
<td>(Const Attr::BlobAttrId &amp;attrId) const</td>
</tr>
</tbody>
</table>
get blob attribute on the entity

void **AddString** (const Attr::StringAttrId &attrId)
add string attribute if not exists

void **AddInt** (const Attr::IntAttrId &attrId)
add int attribute if not exists

void **AddFloat** (const Attr::FloatAttrId &attrId)
add float attribute if not exists

void **AddBool** (const Attr::BoolAttrId &attrId)
add bool attribute if not exists

void **AddFloat4** (const Attr::Float4AttrId &attrId)
add float4 attribute if not exists

void **AddMatrix44** (const Attr::Matrix44AttrId &attrId)
add matrix44 attribute if not exists

void **AddGuid** (const Attr::GuidAttrId &attrId)
add guid attribute if not exists

void **AddBlob** (const Attr::BlobAttrId &attrId)
add blob attribute if not exists

int **GetRefCount** () const
get the current refcount

void **AddRef** ()
increment refcount by one

void **Release** ()
decrement refcount and destroy object if refcount is zero

bool **IsInstanceOf** (const Rtti &rtti) const
return true if this object is instance of given class

bool **IsInstanceOf** (const Util::String &className) const
return true if this object is instance of given class by string

bool **IsInstanceOf** (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool **IsA** (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool **IsA** (const Util::String &rttiName) const
return true if this object is instance of given class, or a
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DumpRefCountingLeaks()</code></td>
<td>Dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void Game::Entity::RegisterPropertyCallback(
    const Ptr< Property >& prop,
    Property::CallbackType callbackType
)
```

register a property callback, called by Property::SetupCallback() method

This method is called from within Property::SetupCallbacks() to register per-frame callback methods with the entity.

```cpp
void Game::Entity::OnActivate()
```

called when attached to world

Called when the game entity has been attached to a game level object. This will attach contained subsystem entities to their respective subsystems.

**Parameters:**
- `l` pointer to a level object to which the entity was attached

- 01-Apr-05 floh graphics entity now created after physics entity

```cpp
void Game::Entity::OnDeactivate()
```

called when removed from world

Called when the game entity has been removed from the game level object. This will remove any contained subsystem entities from their subsystems.

```cpp
void Game::Entity::OnBeginFrame()
```

called at the beginning of the frame
Called on game entities at the begin of the frame.

```cpp
void Game::Entity::OnMoveBefore()
```
called before movement

Called on game entities before movement.

```cpp
void Game::Entity::OnMoveAfter()
```
called after movement

Called on game entities after movement.

```cpp
void Game::Entity::OnRender()
```
called before rendering

Called on game entities before rendering.

```cpp
void Game::Entity::OnRenderDebug()
```
debug rendering called before rendering

Called on game entities before rendering.

```cpp
void Game::Entity::OnLoseActivity()
```
called if entity loses activity

Called on game entities at the begin of the frame.

```cpp
void Game::Entity::OnGainActivity()
```
called if entity gains activity

Called on game entities if gaining/regaining activity
static void
Game::Entity::OnLoad()

called after loading from database has happened

This method is called after the game world has been loaded from the database. At the time when this method is called all entities in the world have already been created and their attributes have been loaded from the database.

This method reads the entity attributes from the world database. A valid GUID attribute must exist on the entity for identification in the database. After the attributes are loaded from the world database, the Property::OnLoad() method will be called on all attached properties.

static void
Game::Entity::OnStart()

called when the entity starts to live in the complete world

This method is called in 2 cases:

When a level is loaded it is called on all entities after OnLoad when the complete world already exist.

When a entity is created at ruTime (while a level is active) OnStart is called after the entity is attached to level.

static void
Game::Entity::OnSave()

called before saving from database happens

This method writes the current entity attributes back to the world database. At least a valid GUID attribute must exist on the entity for identification. If no entry exists yet in the database for the entity, a new one will be created. Before the attributes are written back to the database, the method Property::OnSave() will be called on each attached property.

static void
const
Game::Entity::SetAttr (Attr::Attribute attr) &

generic attribut setter (slow!)

Generic attribute setter method. This is slower then the specialized attribute setter methods!

Attr::Attribute
Game::Entity::GetAttr (Attr::AttrId attrId) const &

generic attribute getter (slow!)

Generic attribute getter method. This is slower then the specialized attribute getter methods!

int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name
Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Game::FeatureUnit
# include <featureunit.h>

Inheritance diagram for Game::FeatureUnit:
**Detailed Description**

A **FeatureUnit** is an encapsulated feature which can be added to an application. E.g. game features can be core features of Nebula3 like Render or Network, or it can be some of the addons like db or physics.

To add a new feature, derive from this class and add it to the **Game::GameServer** on application or statehandler startup.

The **Game::GameServer** will start, load, save, trigger and close your feature.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FeatureUnit()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~FeatureUnit()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>OnActivate()</code></td>
<td>virtual void called from <code>GameServer::AttachGameFeature()</code></td>
</tr>
<tr>
<td><code>OnDeactivate()</code></td>
<td>virtual void called from <code>GameServer::RemoveGameFeature()</code></td>
</tr>
<tr>
<td><code>IsActive()</code></td>
<td>bool const return <code>true</code> if property is currently active</td>
</tr>
<tr>
<td><code>OnLoad()</code></td>
<td>virtual void called from within <code>GameServer::Load()</code> after attributes are loaded</td>
</tr>
<tr>
<td><code>OnStart()</code></td>
<td>virtual void called from within <code>GameServer::OnStart()</code> after <code>OnLoad</code> when the complete world exist</td>
</tr>
<tr>
<td><code>OnSave()</code></td>
<td>virtual void called from within <code>GameServer::Save()</code> before attributes are saved back to database</td>
</tr>
<tr>
<td><code>OnBeginFrame()</code></td>
<td>virtual void called on begin of frame</td>
</tr>
<tr>
<td><code>OnFrame()</code></td>
<td>virtual void called in the middle of the feature trigger cycle</td>
</tr>
<tr>
<td><code>OnEndFrame()</code></td>
<td>virtual void called at the end of the feature trigger cycle</td>
</tr>
<tr>
<td><code>StartRenderDebug()</code></td>
<td>virtual void start render debug</td>
</tr>
<tr>
<td><code>OnRenderDebug()</code></td>
<td>virtual void called when game debug visualization is on</td>
</tr>
<tr>
<td><code>StopRenderDebug()</code></td>
<td>virtual void stop render debug</td>
</tr>
<tr>
<td><code>AttachManager(const Ptr&lt; Manager&gt; &amp;manager)</code></td>
<td>virtual void attach a manager to the game world</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>virtual void RemoveManager (const Ptr&lt; Manager &gt;&amp;manager)</code></td>
<td>remove a manager from the game world</td>
</tr>
<tr>
<td><code>void SetCmdLineArgs (const Util::CommandLineArgs &amp;a)</code></td>
<td>set command line args</td>
</tr>
<tr>
<td><code>const Util::CommandLineArgs &amp; GetCmdLineArgs () const</code></td>
<td>get command line args</td>
</tr>
<tr>
<td><code>void SetRenderDebug (bool b)</code></td>
<td>set flag for rendering debug information</td>
</tr>
<tr>
<td><code>bool HasRenderDebug () const</code></td>
<td>get flag for rendering debug information</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>Class</td>
<td>Function</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td><code>Util::String &amp;</code></td>
<td><code>GetClassName () const</code></td>
</tr>
<tr>
<td><code>Util::FourCC</code></td>
<td><code>GetClassFourCC () const</code></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Game::FeatureUnit::OnActivate( ) [virtual]
called from GameServer::AttachGameFeature()
This method is called by Game::GameServer::ActivateProperties(). Use this method for one-time initializations of the FeatureUnit.
Reimplemented in ScriptFeature::ScriptFeatureUnit.
```

```cpp
void Game::FeatureUnit::OnDeactivate( ) [virtual]
called from GameServer::RemoveGameFeature()
This method is called by Game::GameServer::DeactivateProperties(). Use this method to cleanup stuff which has been initialized in OnActivate().
Reimplemented in ScriptFeature::ScriptFeatureUnit.
```

```cpp
void Game::FeatureUnit::OnLoad( ) [virtual]
called from within GameServer::Load() after attributes are loaded
This method is called from within Game::GameServer::Load() on load of a savegame.
```

```cpp
void Game::FeatureUnit::OnStart( ) [virtual]
called from within GameServer::OnStart() after OnLoad when the complete world exist
This method is called from within Game::GameServer::OnStart(). Its called after all game features are activated and have initialized their subsystems. Use this
void Game::FeatureUnit::OnSave( ) [virtual]
called from within GameServer::Save() before attributes are saved back to database
This method is called from within Game::GameServer::Save(). It's called on save of a game.

void Game::FeatureUnit::OnBeginFrame( ) [virtual]
called on begin of frame
This method is called from Game::GameServer::OnBeginFrame() on all game features attached to an GameServer in the order of attachment. Override this method if your FeatureUnit has to do any work at the beginning of the frame.

void Game::FeatureUnit::OnFrame( ) [virtual]
called in the middle of the feature trigger cycle
This method is called from Game::GameServer::OnMoveBefore() on all game features attached to an GameServer in the order of attachment. Override this method if your FeatureUnit has any work to do before the physics subsystem is triggered.

void Game::FeatureUnit::OnEndFrame( ) [virtual]
called at the end of the feature trigger cycle
This method is called from Game::GameServer::OnRender() on all game features attached to an GameServer in the order of attachment. Override this method if your FeatureUnit has any work to do before rendering happens.

void Game::FeatureUnit::OnRenderDebug( ) [virtual]
called when game debug visualization is on

This method is called from Game::GameServer::OnRenderDebug() on all game features attached to an GameServer in the order of attachment. It's meant for debug issues. It will be called when debug mode is enabled.

```cpp
void Game::FeatureUnit::AttachManager(
    const Ptr<Manager> & manager ) [virtual]
```

attach a manager to the game world

Attach a manager object to the game world. The manager's OnActivate() method will be called once right away, and then its OnFrame() method once per frame.

```cpp
void Game::FeatureUnit::RemoveManager(
    const Ptr<Manager> & manager ) [virtual]
```

remove a manager from the game world

Remove a manager object from the game world. The manager's OnDeactivate() method will be called.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void () [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```
const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Game::GameServer
Game::GameServer Class Reference

#include <gameserver.h>

Inheritance diagram for Game::GameServer:

```
Core::RefCounted

Game::GameServer
```

Detailed Description

The game server setups and runs the game world, consisting of a number of active "game entities". Functionality and queries on the game world are divided amongst several Game Features. This keeps the game server's interface small and clean, and lets Mangalore applications easily extend functionality by implementing new, or deriving from existing game features.

To add or replace FeatureUnit objects, derive from Game::FeatureUnit and add your features on application start or gamestatehandler enter.

The GameServer triggers all attached features. Start and Stop is called within the gamestatehandler to allow all features do stuff after everything is loaded and initialized. Load and Save is invoked from the BaseGameFeature which allows begining a new game, load or save a game.

(C) 2007 RadonLabs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GameServer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~GameServer ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual bool Open ()</td>
<td>Open the game world</td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td>Close the game world</td>
</tr>
<tr>
<td>virtual bool Start ()</td>
<td>Start the game world</td>
</tr>
<tr>
<td>bool HasStarted () const</td>
<td>Has the game world already started</td>
</tr>
<tr>
<td>virtual void Stop ()</td>
<td>Stop the game world</td>
</tr>
<tr>
<td>virtual void OnFrame ()</td>
<td>Trigger the game world</td>
</tr>
<tr>
<td>virtual void NotifyGameLoad ()</td>
<td>Call OnLoad on all game features</td>
</tr>
<tr>
<td>virtual void NotifyGameSave ()</td>
<td>Call OnSave on all game features</td>
</tr>
<tr>
<td>void AttachGameFeature (const Ptr&lt; FeatureUnit &gt; &amp;feature)</td>
<td>Add game feature</td>
</tr>
<tr>
<td>void RemoveGameFeature (const Ptr&lt; FeatureUnit &gt; &amp;feature)</td>
<td>Remove game feature</td>
</tr>
<tr>
<td>bool IsFeatureAttached (const Util::String &amp;stringName) const</td>
<td>Is feature attached</td>
</tr>
<tr>
<td>bool IsQuitRequested ()</td>
<td>Is quit requested</td>
</tr>
<tr>
<td>void SetQuitRequested ()</td>
<td>Request quit</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>int</td>
<td><strong>GetRefCount () const</strong></td>
</tr>
<tr>
<td>void</td>
<td><strong>AddRef ()</strong></td>
</tr>
<tr>
<td>void</td>
<td><strong>Release ()</strong></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf (const Util::String &amp;className) const</strong></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA (const Rtti &amp;rtti) const</strong></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA (const Util::String &amp;rttiName) const</strong></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName () const</strong></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC () const</strong></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

```c++
static void DumpRefCountingLeaks ()

\textit{dump refcounting leaks, call at end of application (NEBULA3\_DEBUG builds only!)}
```
Protected Member Functions

void CheckDebugRendering ()
    check input for render debug and return feature for rendering
Member Function Documentation

```cpp
bool Game::GameServer::Open() [virtual]
```

open the game world

Initialize the game server object. This will create and initialize all subsystems.

```cpp
void Game::GameServer::Close() [virtual]
```

close the game world

Close the game server object.

```cpp
bool Game::GameServer::Start() [virtual]
```

start the game world

Start the game world, called after loading has completed.

```cpp
void Game::GameServer::Stop() [virtual]
```

stop the game world

Stop the game world, called before the world(current level) is cleaned up.

```cpp
void Game::GameServer::OnFrame() [virtual]
```

trigger the game world

Trigger the game server. If your application introduces new or different manager objects, you may also want to override the `Game::GameServer::Trigger()` method if those gameFeatures need
per-frame callbacks.

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Game::Manager
Game::Manager Class Reference

#include <manager.h>

Inheritance diagram for Game::Manager:
Detailed Description

Managers are Singleton objects which care about some "specific global stuff". They should be subclassed by Mangalore applications to implement globals aspects of the application (mainly global game play related stuff).

Managers are created and triggered by game features. The frame trigger functions are invoked when the gameserver triggers the game feature.

Standard-Mangalore uses several Managers to offer timing information (TimeManager), create entities and properties (FactoryManager), manage game entities (EntityManager) and so forth.

Managers are derived from Messaging::Port, so you *can* optionally use them to receive and process messages.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Manager ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~Manager ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <code>OnActivate ()</code></td>
<td>Called when attached to game server</td>
</tr>
<tr>
<td>virtual void <code>OnDeactivate ()</code></td>
<td>Called when removed from game server</td>
</tr>
<tr>
<td>bool <code>IsActive ()</code> const</td>
<td>Return true if currently active</td>
</tr>
<tr>
<td>virtual void <code>OnBeginFrame ()</code></td>
<td>Called before frame by the game server</td>
</tr>
<tr>
<td>virtual void <code>OnFrame ()</code></td>
<td>Called per-frame by the game server</td>
</tr>
<tr>
<td>virtual void <code>OnEndFrame ()</code></td>
<td>Called after frame by the game server</td>
</tr>
<tr>
<td>virtual void <code>OnLoad ()</code></td>
<td>Called after loading game state</td>
</tr>
<tr>
<td>virtual void <code>OnSave ()</code></td>
<td>Called before saving game state</td>
</tr>
<tr>
<td>virtual void <code>OnStart ()</code></td>
<td>Called by <code>Game::Server::Start()</code> when the world is started</td>
</tr>
<tr>
<td>virtual void <code>OnRenderDebug ()</code></td>
<td>Render a debug visualization</td>
</tr>
<tr>
<td>virtual void <code>HandleMessage (const Ptr&lt; Messaging::Message &gt; &amp;msg)</code></td>
<td>Handle a single message (distribute to ports which accept the message)</td>
</tr>
<tr>
<td>void <code>AttachPort (const Ptr&lt; Port &gt; &amp;port)</code></td>
<td>Attach a message port</td>
</tr>
<tr>
<td>void <code>RemovePort (const Ptr&lt; Port &gt; &amp;port)</code></td>
<td>Remove a message port</td>
</tr>
<tr>
<td>bool <code>HasPort (const Ptr&lt; Port &gt; &amp;port)</code> const</td>
<td>Check if a message port exists</td>
</tr>
</tbody>
</table>
virtual void SetupAcceptedMessages ()
override to register accepted messages

void AttachHandler (const Ptr< Handler > &h)
attach a message handler to the port

void RemoveHandler (const Ptr< Handler > &h)
remove a message handler from the port

void RemoveAllHandlers ()
remove all message handler from the port

SizeT GetNumHandlers () const
return number of handlers attached to the port

const Ptr< Handler > & GetHandlerAtIndex (IndexT i) const
get a message handler by index

virtual void Send (const Ptr< Message > &msg)
send a message to the port

const Util::Array< const Id * > & GetAcceptedMessages () const
get the array of accepted messages (sorted)

bool AcceptsMessage (const Id &msgId) const
return true if port accepts this msg

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

`void RegisterMessage (const Id &msgId)`

*register a single accepted message*
Member Function Documentation

void Game::Manager::OnActivate() [virtual]
called when attached to game server

This method is called when the manager is attached to the game server. The manager base class will register its message port with the message server.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager, BaseGameFeature::EnvQueryManager, BaseGameFeature::GlobalAttrsManager, Script::DialogManager, and Script::ScriptManager.

void Game::Manager::OnDeactivate() [virtual]
called when removed from game server

This method is called when the manager is removed from the game server. It will unregister its message port from the message server at this point.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager, BaseGameFeature::EntityManager, BaseGameFeature::EnvEntityManager, BaseGameFeature::EnvQueryManager, Script::DialogManager, and Script::ScriptManager.

void Game::Manager::OnBeginFrame() [virtual]
called before frame by the game server

Called before frame, override in subclasses
Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Game::Manager::OnEndFrame() [virtual]
```
called after frame by the game server

Called after frame, override in subclasses

Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Messaging::Dispatcher::HandleMessage(const Messaging::Message &msg) [virtual,inherited]
```
handle a single message (distribute to ports which accept the message)

Handle a message. The message will only be distributed to ports which accept the message.

Reimplemented from **Messaging::Port**.

Reimplemented in **Script::DialogManager**.

```cpp
void Messaging::Dispatcher::AttachPort(const Port &port) [inherited]
```
attach a message port

Attach a new message port.

**Parameters:**

- `port` pointer to a message port object

```cpp
void Messaging::Dispatcher::RemovePort(const Port &port) [inherited]
```
remove a message port
Remove a message port object.

**Parameters:**

`handler` pointer to message port object to be removed

```cpp
bool Messaging::Dispatcher::HasPort(Ptr<Port> & port) const [inherited]
return true if a port exists
```

Return true if a port is already attached.

```cpp
void Messaging::Port::AttachHandler(Ptr<Handler> & h) [inherited]
attach a message handler to the port
```

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler(Ptr<Handler> & h) [inherited]
remove a message handler from the port
```

Remove a message handler from the port.

```cpp
void Messaging::Port::Send(Ptr<Message> & msg) [virtual, inherited]
send a message to the port
```

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.
int Core::RefCounted::GetRefCount(  ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef(  ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release(  ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName(  ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC(  ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks(  ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Game::PhysicsFeatureUnit
Game::PhysicsFeatureUnit Class Reference

#include <physicsfeatureunit.h>
Detailed Description

The physics feature offers the use of collision and physical representation of game objects.

It uses the physics addon with use of a third party physics system (like ODE).

It offers a collide property for static environment objects and a physics property for passiv moving dynamic objects and an actorphysics property for active dynamic moving objects.

Additional there is a mouse gripper property for grabbing and throwing physics object around.

IMPORTANT: the physics server and so the physics simulation system is triggered in OnFrame of the feature so

(C) 2007 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Game::Property
Game::Property Class Reference

#include <property.h>

Inheritance diagram for Game::Property:
Detailed Description

Properties are attached to game entities to add specific functionality or behaviours to the entity. For instance, you add a GraphicsProperty if the entity should be able render itself, or you add an AudioProperty if the entity should be able to emit sound. Properties are derived from the `Messaging::Port` class and as such may receive and handle messages but they are not required to do so.

(C) 2007 Radon Labs GmbH
Public Types

enum CallbackType
callback types
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Property()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~Property()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>const Ptr&lt; Entity &gt; &amp; GetEntity()</code></td>
<td>Get entity this property is attached to</td>
</tr>
<tr>
<td><code>bool HasEntity()</code></td>
<td>Return true if entity pointer is valid</td>
</tr>
<tr>
<td>virtual <code>void SetupDefaultAttributes()</code></td>
<td>Setup the property’s attributes to their default state</td>
</tr>
<tr>
<td>virtual <code>void SetupCallbacks()</code></td>
<td>Setup callbacks for this property, call by entity in <code>OnActivate()</code></td>
</tr>
<tr>
<td>virtual <code>void OnActivate()</code></td>
<td>Called from Entity::ActivateProperties()</td>
</tr>
<tr>
<td>virtual <code>void OnDeactivate()</code></td>
<td>Called from Entity::DeactivateProperties()</td>
</tr>
<tr>
<td><code>bool IsActive()</code></td>
<td>Return true if property is currently active</td>
</tr>
<tr>
<td>virtual <code>void OnLoad()</code></td>
<td>Called from within Entity::Load() after attributes are loaded</td>
</tr>
<tr>
<td>virtual <code>void OnStart()</code></td>
<td>Called from within Entity::OnStart() after OnLoad when the complete world exist</td>
</tr>
<tr>
<td>virtual <code>void OnSave()</code></td>
<td>Called from within Entity::Save() before attributes are saved back to database</td>
</tr>
<tr>
<td>virtual <code>void OnBeginFrame()</code></td>
<td>Called on begin of frame</td>
</tr>
<tr>
<td>virtual <code>void OnMoveBefore()</code></td>
<td>Called before movement happens</td>
</tr>
<tr>
<td>virtual <code>void OnMoveAfter()</code></td>
<td>Called after movement has happened</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void OnRender()</td>
<td>called before rendering happens</td>
</tr>
<tr>
<td>virtual void OnRenderDebug()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void OnLoseActivity()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void OnGainActivity()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void HandleMessage()</td>
<td>(const Ptr&lt;br Message::Message &gt; &amp;msg) handle a single message</td>
</tr>
<tr>
<td>virtual void SetupAcceptedMessages()</td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>void AttachHandler()</td>
<td>(const Ptr&lt;br Handler &gt; &amp;h) attach a message handler to the port</td>
</tr>
<tr>
<td>void RemoveHandler()</td>
<td>(const Ptr&lt;br Handler &gt; &amp;h) remove a message handler from the port</td>
</tr>
<tr>
<td>void RemoveAllHandlers()</td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td>SizeT GetNumHandlers()</td>
<td>const return number of handlers attached to the port</td>
</tr>
<tr>
<td>const Ptr&lt;br Handler &gt; &amp;</td>
<td>GetHandlerAtIndex(IndexT i) const get a message handler by index</td>
</tr>
<tr>
<td>virtual void Send()</td>
<td>(const Ptr&lt;br Message &gt; &amp;msg) send a message to the port</td>
</tr>
<tr>
<td>const Util::Array&lt;br Id * &gt; &amp;</td>
<td>GetAcceptedMessages() const get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td>bool AcceptsMessage(Id &amp;msgId)</td>
<td>const return true if port accepts this msg</td>
</tr>
<tr>
<td>int GetRefCount()</td>
<td>const get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName()</code> const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC()</code> const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

void **SetEntity** (const **Ptr< Entity >** &v)**

*Set entity, this is attached to, to `v`.*

void **ClearEntity** ()

*Remove entity.*

void **RegisterMessage** (const **Id** &msgId)

*register a single accepted message*
Member Function Documentation

```cpp
void Game::Property::SetupDefaultAttributes() [virtual]
```

setup the property's attributes to their default state

If a property adds attributes to an entity, override this method to setup their default state. This method is called before the entity is even initialized from the database. After this method, entity attributes may be overwritten from the database, and after that from a stream.


```cpp
void Game::Property::SetupCallbacks() [virtual]
```

setup callbacks for this property, call by entity in `OnActivate()`

Tells the entity what per-frame callback methods should be called for this property. The method is called after `SetupDefaultAttributes()` by the entity, and the property is expected to call the `Entity::RegisterPropertyCallback()` once for every callback method (`OnBeginFrame()`, `OnMoveBefore()`, ...) that should be called per-frame.

Reimplemented in `GraphicsFeature::CameraProperty`, `GraphicsFeature::GraphicsProperty`,
GraphicsFeature::MayaCameraProperty,
PhysicsFeature::ActorPhysicsProperty,
PhysicsFeature::PhysicsProperty,
PhysicsFeature::TriggerProperty, and
StateObjectFeature::StateProperty.

void
Game::Property::OnActivate() [virtual]
called from Entity::ActivateProperties()

This method is called by Game::Entity::ActivateProperties(). Use this method for one-time initializations of the property.

Reimplemented in GraphicsFeature::ChaseCameraProperty,
GraphicsFeature::GraphicsProperty,
GraphicsFeature::MayaCameraProperty,
PhysicsFeature::ActorPhysicsProperty,
PhysicsFeature::PhysicsProperty,
PhysicsFeature::TriggerProperty, and
StateObjectFeature::StateGraphicsProperty.

void
Game::Property::OnDeactivate() [virtual]
called from Entity::DeactivateProperties()

This method is called by Game::Entity::DeactivateProperties(). Use this method to cleanup stuff which has been initialized in
OnActivate().

Reimplemented in GraphicsFeature::CameraProperty,
GraphicsFeature::GraphicsProperty,
GraphicsFeature::MayaCameraProperty,
PhysicsFeature::ActorPhysicsProperty,
PhysicsFeature::EnvironmentCollideProperty,
PhysicsFeature::PhysicsProperty,
PhysicsFeature::TriggerProperty,
StateObjectFeature::StateGraphicsProperty, and
StateObjectFeature::StateProperty.
void Game::Property::OnLoad ( ) [virtual]

called from within Entity::Load() after attributes are loaded

This method is called from within Game::Entity::Load() after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void Game::Property::OnStart ( ) [virtual]

called from within Entity::OnStart() after OnLoad when the complete world exist

This method is called from within Game::Entity::OnStart(). This is the moment when the world is complete and the entity can establish connections to other entities.

Reimplemented in GraphicsFeature::CameraProperty.

void Game::Property::OnSave ( ) [virtual]

called from within Entity::Save() before attributes are saved back to database

This method is called from within Game::Entity::Save() before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.
void Game::Property::OnBeginFrame() [virtual]
called on begin of frame

This method is called from Game::Entity::OnBeginFrame() on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void Game::Property::OnMoveBefore() [virtual]
called before movement happens

This method is called from Game::Entity::OnMoveBefore() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before the physics subsystem is triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty.

void Game::Property::OnMoveAfter() [virtual]
called after movement has happened

This method is called from Game::Entity::OnMoveAfter() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty, and PhysicsFeature::PhysicsProperty.

void Game::Property::OnRender() [virtual]
called before rendering happens

This method is called from `Game::Entity::OnRender()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before rendering happens.

Reimplemented in `GraphicsFeature::CameraProperty`, `GraphicsFeature::ChaseCameraProperty`, and `GraphicsFeature::MayaCameraProperty`.

```cpp
void Game::Property::OnRenderDebug() [virtual]
```

called when game debug visualization is on

This method is called from `Game::Entity::OnRenderDebug()` on all properties attached to an entity in the order of attachment. It's meant for debug issues. It will be called when debug mode is enabled.

Reimplemented in `GraphicsFeature::GraphicsProperty`, `PhysicsFeature::ActorPhysicsProperty`, and `PhysicsFeature::TriggerProperty`.

```cpp
void Game::Property::OnLoseActivity() [virtual]
```

called when game debug visualization is on

This method is called from `Game::Entity::OnLoseActivity()` on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`.

```cpp
void Game::Property::OnGainActivity() [virtual]
```

called when game debug visualization is on
This method is called from `Game::Entity::OnRenderDebug()` on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

```cpp
void Messaging::Port::AttachHandler ( const Ptr<Handler> & h ) [inherited]
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler ( const Ptr<Handler> & h ) [inherited]
```

remove a message handler from the port

Remove a message handler from the port.

```cpp
void Messaging::Port::Send ( const Ptr<Message> & msg ) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef()
    [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release()
    [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName()
    [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC()
    [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks()
    [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Graphics::AbstractLightEntity
Graphics::AbstractLightEntity Class Reference

#include <abstractlightentity.h>

Inheritance diagram for Graphics::AbstractLightEntity:
Detailed Description

Public interface to InternalGraphics::InternalAbstractLightEntity.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AbstractLightEntity ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~AbstractLightEntity ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>Lighting::LightType::Code <strong>GetLightType () const</strong></td>
<td>Get the light type (defined by subclass)</td>
</tr>
<tr>
<td>void <strong>SetColor (const Math::float4 &amp;c)</strong></td>
<td>Set primary light color</td>
</tr>
<tr>
<td>const Math::float4 &amp; <strong>GetColor () const</strong></td>
<td>Get primary light color</td>
</tr>
<tr>
<td>void <strong>SetColor (bool b)</strong></td>
<td>Enable/disable shadow casting</td>
</tr>
<tr>
<td>bool <strong>GetCastShadows () const</strong></td>
<td>Get shadow casting flag</td>
</tr>
<tr>
<td>void <strong>SetProjMapUvOffsetAndScale (const Math::float4 &amp;v)</strong></td>
<td>Set projection map UV offset and scale (xy-&gt;offset, zw-&gt;scale)</td>
</tr>
<tr>
<td>const Math::float4 &amp; <strong>GetProjMapUvOffsetAndScale () const</strong></td>
<td>Get projection map UV offset and scale</td>
</tr>
<tr>
<td>bool <strong>IsValid () const</strong></td>
<td>Return true if entity is valid (is attached to a stage)</td>
</tr>
<tr>
<td>GraphicsEntityType::Code <strong>GetType () const</strong></td>
<td>Get the entity type</td>
</tr>
<tr>
<td>void <strong>SetTransform (const Math::matrix44 &amp;m)</strong></td>
<td>Set the entity's world space transform</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; <strong>GetTransform () const</strong></td>
<td>Get the entity's world space transform</td>
</tr>
<tr>
<td>void <strong>SetVisible (bool b)</strong></td>
<td>Set the entity's visibility</td>
</tr>
<tr>
<td>bool <strong>isVisible () const</strong></td>
<td>Return true if entity is set to visible</td>
</tr>
<tr>
<td>C++ Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Stage &gt; &amp; GetStage () const</code></td>
<td>get the stage proxy this entity is attached to</td>
</tr>
<tr>
<td>Math::ClipStatus::Type GetClipStatus () const</td>
<td>get current clip status</td>
</tr>
<tr>
<td>const Math::bbox &amp; GetLocalBoundingBox () const</td>
<td>get the local bounding box</td>
</tr>
<tr>
<td>const Math::bbox &amp; GetGlobalBoundingBox () const</td>
<td>get the global bounding box</td>
</tr>
<tr>
<td>Timing::Time GetEntityTime () const</td>
<td>get current entity time</td>
</tr>
<tr>
<td>void SendMsg (const <code>Ptr&lt; GraphicsEntityMessage &gt; &amp;msg</code>)</td>
<td>send a message to the server-side graphics entity</td>
</tr>
<tr>
<td>bool IsObjectRefValid () const</td>
<td>test if the entity's object ref is valid</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Threading::ObjectRef &gt; &amp; GetObjectRef () const</code></td>
<td>get the entity handle</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const <code>Rtti &amp;rtti</code>) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const <code>Util::String &amp;className</code>)</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const <code>Util::FourCC &amp;classFourCC</code>)</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const <code>Rtti &amp;rtti</code>) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>IsA (const <code>Util::String &amp;rttiName</code>)</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>const</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetLightType(Lighting::LightType::Code c)</code></td>
<td>set the light type (must be called from sub-classes constructor)</td>
</tr>
<tr>
<td><code>void SetType(GraphicsEntityType::Code t)</code></td>
<td>set graphics entity type, called from constructor of subclass</td>
</tr>
<tr>
<td><code>virtual void Setup(const Ptr&lt; Stage &gt;&amp; stage)</code></td>
<td>called by stage when entity should setup itself</td>
</tr>
<tr>
<td><code>virtual void Discard()</code></td>
<td>called by stage when entity should discard itself</td>
</tr>
<tr>
<td><code>virtual void OnSetupSharedData()</code></td>
<td>called to setup the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td><code>virtual void OnDiscardSharedData()</code></td>
<td>called to discard the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td><code>void SendCreateMsg(const Ptr&lt; CreateGraphicsEntity &gt;&amp; msg)</code></td>
<td>send off a specific create message from the subclass</td>
</tr>
<tr>
<td><code>virtual void OnTransformChanged()</code></td>
<td>called when transform matrix changed</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Graphics::GraphicsEntity::SendMsg ( const Ptr<GraphicsEntityMessage> msg ) [inherited]

send a message to the server-side graphics entity

Send a generic GraphicsEntityMessage to the server-side entity.

void Graphics::GraphicsEntity::Setup ( const Ptr<Stage> &stage_ ) [protected, virtual, inherited]

called by stage when entity should setup itself

Setup the graphics entity. Subclasses must send a specific creation message in this method. This method is called from StageProxy::AttachEntityProxy().

Reimplemented in Graphics::CameraEntity.

void Graphics::GraphicsEntity::Discard ( ) [protected, virtual, inherited]

called by stage when entity should discard itself

Discard the server-side graphics entity. This method is called from StageProxy::RemoveEntityProxy(). There's special handling if the server-side entity hasn't been created yet, in this case, a pointer to the original create message must be handed over to the render thread.

Reimplemented in Graphics::CameraEntity.

void Graphics::GraphicsEntity::OnSetupSharedData ( ) [protected, virtual, inherited]

called to setup the FrameSyncSharedData object of the entity

Setup the shared data object. If subclasses wish to add more
members to the shared data object, they must subclass from InternalGraphics::InternalGraphicsEntitySharedData, setup a FrameSyncSharedData object with this data type, and NOT CALL the parent class method! So the lowest subclass defines the type of the shared data!

```cpp
void Graphics::GraphicsEntity::OnDiscardSharedData() [protected, virtual, inherited]
```
called to discard the FrameSyncSharedData object of the entity

Discard the shared data object. See `OnSetupSharedData()` for details!

```cpp
void Graphics::GraphicsEntity::SendCreateMsg(const Ptr<CreateGraphicsEntity msg> &)
```
send off a specific create message from the subclass

This method must be called from the `Setup()` method of a subclass to send off a specific creation message. The message will be stored in the proxy to get the entity handle back later when the server-side graphics entity has been created.

```cpp
void Graphics::GraphicsEntity::OnTransformChanged() [protected, virtual, inherited]
```
called when transform matrix changed

Called by `SetTransform()`. This gives subclasses a chance to react to changes on the transformation matrix.

Reimplemented in `Graphics::CameraEntity`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

const Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Graphics::CameraEntity
Graphics::CameraEntity Class Reference

```cpp
#include <cameraentity.h>
```

Inheritance diagram for Graphics::CameraEntity:

```
Core::RefCounted
```
```
Graphics::GraphicsEntity
```
```
Graphics::CameraEntity
```
Detailed Description

Client-side proxy of a `InternalGraphics::InternalCameraEntity`. NOTE: all getter-methods of this class return client-side cached values, not the actual server-side values. Thus they may be off by some amount, since the render thread may run at a different frame rate than the client thread!

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CameraEntity()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~CameraEntity()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>bool IsAttachedToView()</code></td>
<td>Return true if this camera is currently attached to a view</td>
</tr>
<tr>
<td><code>const Ptr&lt; View &gt; &amp; GetView()</code></td>
<td>Get the view this camera is currently attached to</td>
</tr>
<tr>
<td><code>void SetCameraSettings(const Shared::CameraSettings &amp;camSettings)</code></td>
<td>Set new camera settings</td>
</tr>
<tr>
<td><code>const Shared::CameraSettings &amp; GetCameraSettings()</code></td>
<td>Get camera settings</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetProjTransform()</code></td>
<td>Get projection matrix</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetViewTransform()</code></td>
<td>Get view transform (inverse transform)</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetViewProjTransform()</code></td>
<td>Get view projection matrix</td>
</tr>
<tr>
<td><code>const Math::frustum &amp; GetViewFrustum()</code></td>
<td>Get view frustum</td>
</tr>
<tr>
<td><code>bool IsValid()</code></td>
<td>Return true if entity is valid (is attached to a stage)</td>
</tr>
<tr>
<td><code>GraphicsEntityType::Code GetType()</code></td>
<td>Get the entity type</td>
</tr>
<tr>
<td><code>void SetTransform(const Math::matrix44 &amp;m)</code></td>
<td>Set the entity's world space transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetTransform()</code></td>
<td>Get the entity's world space transform</td>
</tr>
<tr>
<td><code>void SetVisible(bool b)</code></td>
<td>Set the entity's visibility</td>
</tr>
<tr>
<td>Member Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>isVisible()</code> const</td>
<td>return true if entity is set to visible</td>
</tr>
<tr>
<td><code>Ptr&lt; Stage &gt; &amp; GetStage()</code> const</td>
<td>get the stage proxy this entity is attached to</td>
</tr>
<tr>
<td><code>Math::ClipStatus::Type GetClipStatus()</code> const</td>
<td>get current clip status</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetLocalBoundingBox()</code> const</td>
<td>get the local bounding box</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetGlobalBoundingBox()</code> const</td>
<td>get the global bounding box</td>
</tr>
<tr>
<td><code>Timing::Time GetEntityTime()</code> const</td>
<td>get current entity time</td>
</tr>
<tr>
<td><code>void SendMsg(const Ptr&lt; GraphicsEntityMessage &gt; &amp;msg)</code></td>
<td>send a message to the server-side graphics entity</td>
</tr>
<tr>
<td><code>isObjectRefValid()</code> const</td>
<td>test if the entity's object ref is valid</td>
</tr>
<tr>
<td><code>Ptr&lt; Threading::ObjectRef &gt; &amp; GetObjectRef()</code> const</td>
<td>get the entity handle</td>
</tr>
<tr>
<td><code>int GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti)</code> const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool IsA(const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA(const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName() const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC() const</td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>static void DumpRefCountingLeaks ()</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td></td>
</tr>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
<td></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual void Setup (const Ptr&lt; Stage &gt; &amp;stage)</code></td>
<td>called by stage when entity should setup itself</td>
</tr>
<tr>
<td><code>virtual void Discard ()</code></td>
<td>called by stage when entity should discard itself</td>
</tr>
<tr>
<td><code>virtual void OnTransformChanged ()</code></td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td><code>void OnAttachToView (const Ptr&lt; View &gt; &amp;view)</code></td>
<td>called by view when this camera entity should become the view's camera</td>
</tr>
<tr>
<td><code>void OnRemoveFromView (const Ptr&lt; View &gt; &amp;view)</code></td>
<td>called by view when this camera is no longer the view's camera</td>
</tr>
<tr>
<td><code>void SetType (GraphicsEntityType::Code t)</code></td>
<td>set graphics entity type, called from constructor of subclass</td>
</tr>
<tr>
<td><code>virtual void OnSetupSharedData ()</code></td>
<td>called to setup the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td><code>virtual void OnDiscardSharedData ()</code></td>
<td>called to discard the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td><code>void SendCreateMsg (const Ptr&lt; CreateGraphicsEntity &gt; &amp;msg)</code></td>
<td>send off a specific create message from the subclass</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Graphics::CameraEntity::SetCameraSettings ( const Shared::CameraSettings& camSettings )
```

set new camera settings

Set new the camera settings. This updates the internal matrices.

```cpp
void Graphics::CameraEntity::Setup ( const Ptr< Stage >& stage_ ) [protected, virtual]
```

called by stage when entity should setup itself

Setup the server-side camera entity.

Reimplemented from **Graphics::GraphicsEntity**.

```cpp
void Graphics::CameraEntity::Discard ( ) [protected, virtual]
```

called by stage when entity should discard itself

Property remove from view if we are attached to a view.

Reimplemented from **Graphics::GraphicsEntity**.

```cpp
void Graphics::CameraEntity::OnTransformChanged ( ) [protected, virtual]
```

called when transform matrix changed

We need to keep track of modifications of the transformation matrix.

Reimplemented from **Graphics::GraphicsEntity**.
send a message to the server-side graphics entity

Send a generic GraphicsEntityMessage to the server-side entity.

```cpp
void Graphics::GraphicsEntity::OnSetupSharedData() [protected, virtual, inherited]
```
called to setup the FrameSyncSharedData object of the entity

Setup the shared data object. If subclasses wish to add more members to the shared data object, they must subclass from InternalGraphics::InternalGraphicsEntitySharedData, setup a FrameSyncSharedData object with this data type, and NOT CALL the parent class method! So the lowest subclass defines the type of the shared data!

```cpp
void Graphics::GraphicsEntity::OnDiscardSharedData() [protected, virtual, inherited]
```
called to discard the FrameSyncSharedData object of the entity

Discard the shared data object. See `OnSetupSharedData()` for details!

```cpp
void Graphics::GraphicsEntity::SendCreateMsg(const Ptr<CreateGraphicsEntity> msg) [protected, inherited]
```
send off a specific create message from the subclass

This method must be called from the `Setup()` method of a subclass to send off a specific creation message. The message will be stored in the proxy to get the entity handle back later when the server-side graphics entity has been created.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Graphics::Display
Graphics::Display Class Reference

#include <display.h>

Inheritance diagram for Graphics::Display:

```
Core::RefCounted
  |
  v
Graphics::Display
```
Detailed Description

The **Display** object is used to access DisplayDevice functionality from a different thread. Usually only the main thread creates a display object.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Display ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~Display ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>DisplaySettings &amp; Settings ()</strong></td>
<td>access to display settings</td>
</tr>
<tr>
<td>void *<em>SetParentWindow (void <em>h)</em></em></td>
<td>set optional parent window handle (a HWND if running under windows)</td>
</tr>
<tr>
<td>void * <strong>GetParentWindow () const</strong></td>
<td>get optional parent window handle</td>
</tr>
<tr>
<td>void <strong>SetResourceMappers (const Util::Array&lt; Resources::ResourceMapper &gt; &amp;resourceMappers)</strong></td>
<td>set resource mappers (NOTE: there is no Getter for this to prevent messing around with render-thread objects!)</td>
</tr>
<tr>
<td>void <strong>Open ()</strong></td>
<td>open the display (waits for completion)</td>
</tr>
<tr>
<td>void <strong>Close ()</strong></td>
<td>close the display (waits for completion)</td>
</tr>
<tr>
<td>bool <strong>IsOpen () const</strong></td>
<td>return true if display is currently open</td>
</tr>
<tr>
<td>bool <strong>AdapterExists (CoreGraphics::Adapter::Code adapter)</strong></td>
<td>return true if adapter exists (waits for completion)</td>
</tr>
<tr>
<td><strong>Util::Array&lt; CoreGraphics::DisplayMode &gt; GetAvailableDisplayModes (CoreGraphics::Adapter::Code adapter, CoreGraphics::PixelFormat::Code pixelFormat)</strong></td>
<td>get available display modes on given adapter (waits for completion)</td>
</tr>
<tr>
<td>bool <strong>SupportsDisplayMode (CoreGraphics::Adapter::Code adapter, CoreGraphics::DisplayMode &amp;requestedMode)</strong></td>
<td>return true if a given display mode is supported (waits for completion)</td>
</tr>
<tr>
<td>Class</td>
<td>Function</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>CoreGraphics::DisplayMode</td>
<td>GetCurrentAdapterDisplayMode</td>
</tr>
<tr>
<td>CoreGraphics::AdapterInfo</td>
<td>GetAdapterInfo</td>
</tr>
<tr>
<td></td>
<td>AttachDisplayEventHandler</td>
</tr>
<tr>
<td></td>
<td>RemoveDisplayEventHandler</td>
</tr>
<tr>
<td></td>
<td>AttachRenderEventHandler</td>
</tr>
<tr>
<td></td>
<td>RemoveRenderEventHandler</td>
</tr>
<tr>
<td></td>
<td>GetRefCount</td>
</tr>
<tr>
<td></td>
<td>AddRef</td>
</tr>
<tr>
<td></td>
<td>Release</td>
</tr>
<tr>
<td></td>
<td>IsInstanceOf</td>
</tr>
<tr>
<td></td>
<td>IsInstanceOf</td>
</tr>
<tr>
<td></td>
<td>IsInstanceOf</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>`IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class,</td>
</tr>
<tr>
<td>bool `IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, class, by string</td>
</tr>
<tr>
<td>bool `IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; `GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC </code>GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Graphics::DisplaySettings
Graphics::DisplaySettings Class Reference

#include <displaysettings.h>
Detailed Description

Wraps display settings into a simple object.

(C) 2009 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DisplaySettings()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>CoreGraphics::DisplayMode &amp; DisplayMode()</code></td>
<td>DisplayMode accessor.</td>
</tr>
<tr>
<td><code>const CoreGraphics::DisplayMode &amp; DisplayMode()</code> const</td>
<td>const DisplayMode accessor</td>
</tr>
<tr>
<td><code>void SetAdapter(CoreGraphics::Adapter::Code a)</code></td>
<td>set display adapter to use</td>
</tr>
<tr>
<td><code>CoreGraphics::Adapter::Code GetAdapter()</code> const</td>
<td>get display adapter</td>
</tr>
<tr>
<td><code>void SetAntiAliasQuality(CoreGraphics::AntiAliasQuality::Code aa)</code></td>
<td>set antialias quality</td>
</tr>
<tr>
<td><code>CoreGraphics::AntiAliasQuality::Code GetAntiAliasQuality()</code> const</td>
<td>get antialias quality</td>
</tr>
<tr>
<td><code>void SetFullscreen(bool b)</code></td>
<td>set windowed/fullscreen mode</td>
</tr>
<tr>
<td><code>bool IsFullscreen()</code> const</td>
<td>get windowed/fullscreen mode</td>
</tr>
<tr>
<td><code>void SetDisplayModeSwitchEnabled(bool b)</code></td>
<td>enable display mode switch when running fullscreen (default is true);</td>
</tr>
<tr>
<td><code>bool IsDisplayModeSwitchEnabled()</code> const</td>
<td>is display mode switch enabled for fullscreen?</td>
</tr>
<tr>
<td><code>void SetTripleBufferingEnabled(bool b)</code></td>
<td>enable triple buffer for fullscreen (default is double buffering)</td>
</tr>
<tr>
<td><code>bool IsTripleBufferingEnabled()</code> const</td>
<td>is triple buffer enabled for fullscreen?</td>
</tr>
<tr>
<td><code>void SetAlwaysOnTop(bool b)</code></td>
<td>set always-on-top behaviour</td>
</tr>
<tr>
<td><code>bool IsAlwaysOnTop()</code> const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void SetVerticalSyncEnabled(bool b)</code></td>
<td>turn vertical sync on/off</td>
</tr>
<tr>
<td><code>bool IsVerticalSyncEnabled()</code></td>
<td>get vertical sync flag</td>
</tr>
<tr>
<td><code>void SetIconName(const Util::String &amp;s)</code></td>
<td>set optional window icon resource name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetIconName()</code></td>
<td>get optional window icon resource name</td>
</tr>
<tr>
<td><code>void SetWindowTitle(const Util::String &amp;s)</code></td>
<td>set optional window title</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetWindowTitle()</code></td>
<td>get optional window title</td>
</tr>
</tbody>
</table>
Graphics::GlobalLightEntity
Graphics::GlobalLightEntity Class Reference

#include <globallightentity.h>

Inheritance diagram for Graphics::GlobalLightEntity:
Detailed Description

Client-side proxy of a InternalGraphics::InternalGlobalLightEntity.

(C) 2008 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GlobalLightEntity ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>void SetBackLightColor (const Math::float4 &amp;c)</code></td>
<td>set the backlight color</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetBackLightColor () const</code></td>
<td>get the backlight color</td>
</tr>
<tr>
<td><code>void SetAmbientLightColor (const Math::float4 &amp;val)</code></td>
<td>set AmbientLightColor</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetAmbientLightColor () const</code></td>
<td>get AmbientLightColor</td>
</tr>
<tr>
<td><code>void SetBackLightOffset (float val)</code></td>
<td>set BackLightOffset</td>
</tr>
<tr>
<td><code>float GetBackLightOffset () const</code></td>
<td>get BackLightOffset</td>
</tr>
<tr>
<td><code>void SetAllParams (const Math::float4 &amp;color, const Math::float4 &amp;backColor, const Math::float4 &amp;ambColor, float backOffset, bool castShadows)</code></td>
<td>set all params (only sends 1 msg to render thread instead of 4)</td>
</tr>
<tr>
<td><code>Lighting::LightType::Code GetLightType () const</code></td>
<td>get the light type (defined by subclass)</td>
</tr>
<tr>
<td><code>void SetColor (const Math::float4 &amp;c)</code></td>
<td>set primary light color</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetColor () const</code></td>
<td>get primary light color</td>
</tr>
<tr>
<td><code>void SetCastShadows (bool b)</code></td>
<td>enable/disable shadow casting</td>
</tr>
<tr>
<td><code>bool GetCastShadows () const</code></td>
<td>get shadow casting flag</td>
</tr>
<tr>
<td><code>SetProjMapUvOffsetAndScale</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void (const Math::float4 &amp;v)</code></td>
<td>set projection map UV offset and scale (xy-&gt;offset, zw-&gt;scale)</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetProjMapUvOffsetAndScale()</code></td>
<td>get projection map UV offset and scale</td>
</tr>
<tr>
<td><code>bool IsValid () const</code></td>
<td>return true if entity is valid (is attached to a stage)</td>
</tr>
<tr>
<td><code>GraphicsEntityType::Code GetTypeDef () const</code></td>
<td>get the entity type</td>
</tr>
<tr>
<td><code>void SetTransform (const Math::matrix44 &amp;m)</code></td>
<td>set the entity's world space transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetTransform () const</code></td>
<td>get the entity's world space transform</td>
</tr>
<tr>
<td><code>void SetVisible (bool b)</code></td>
<td>set the entity's visibility</td>
</tr>
<tr>
<td><code>bool IsVisible () const</code></td>
<td>return true if entity is set to visible</td>
</tr>
<tr>
<td><code>const Ptr&lt; Stage &gt; &amp; GetStage () const</code></td>
<td>get the stage proxy this entity is attached to</td>
</tr>
<tr>
<td><code>Math::ClipStatus::Type GetClipStatus () const</code></td>
<td>get current clip status</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetLocalBoundingBox () const</code></td>
<td>get the local bounding box</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetGlobalBoundingBox () const</code></td>
<td>get the global bounding box</td>
</tr>
<tr>
<td><code>Timing::Time GetEntityTime () const</code></td>
<td>get current entity time</td>
</tr>
<tr>
<td><code>void SendMsg (const Ptr&lt; GraphicsEntityMessage &gt; &amp;msg)</code></td>
<td>send a message to the server-side graphics entity</td>
</tr>
<tr>
<td><code>bool IsObjectRefValid () const</code></td>
<td>test if the entity's object ref is valid</td>
</tr>
<tr>
<td><code>constPtr&lt; Threading::ObjectRef &gt; &amp; GetObjectRef () const</code></td>
<td>get the entity handle</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

- **GetClassName** () const
  - get the class name

- **GetClassFourCC** () const
  - get the class **FourCC** code
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetLightType (Lighting::LightType::Code c)</code></td>
<td>set the light type (must be called from sub-classes constructor)</td>
</tr>
<tr>
<td><code>void SetType (GraphicsEntityType::Code t)</code></td>
<td>set graphics entity type, called from constructor of subclass</td>
</tr>
<tr>
<td><code>virtual void Discard ()</code></td>
<td>called by stage when entity should discard itself</td>
</tr>
<tr>
<td><code>virtual void OnSetupSharedData ()</code></td>
<td>called to setup the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td><code>virtual void OnDiscardSharedData ()</code></td>
<td>called to discard the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td><code>void SendCreateMsg (const Ptr&lt; CreateGraphicsEntity &gt; &amp;msg)</code></td>
<td>send off a specific create message from the subclass</td>
</tr>
<tr>
<td><code>virtual void OnTransformChanged ()</code></td>
<td>called when transform matrix changed</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Graphics::GraphicsEntity::SendMsg ( const Ptr< GraphicsEntityMessage > msg ) [inherited]
```

send a message to the server-side graphics entity

Send a generic GraphicsEntityMessage to the server-side entity.

```cpp
void Graphics::GraphicsEntity::Discard ( ) [protected, virtual, inherited]
```

called by stage when entity should discard itself

Discard the server-side graphics entity. This method is called from StageProxy::RemoveEntityProxy(). There's special handling if the server-side entity hasn't been created yet, in this case, a pointer to the original create message must be handed over to the render thread.

Reimplemented in Graphics::CameraEntity.

```cpp
void Graphics::GraphicsEntity::OnSetupSharedData ( ) [protected, virtual, inherited]
```

called to setup the FrameSyncSharedData object of the entity

Setup the shared data object. If subclasses wish to add more members to the shared data object, they must subclass from InternalGraphics::InternalGraphicsEntitySharedData, setup a FrameSyncSharedData object with this data type, and NOT CALL the parent class method! So the lowest subclass defines the type of the shared data!

```cpp
void Graphics::GraphicsEntity::OnDiscardSharedData ( ) [protected, virtual, inherited]
```

called to discard the FrameSyncSharedData object of the entity

Discard the shared data object. See OnSetupSharedData() for
details!

```cpp
void Graphics::GraphicsEntity::SendCreateMsg(const Ptr<CreateGraphicsEntity> msg) [protected, inherited]
```

send off a specific create message from the subclass

This method must be called from the `Setup()` method of a subclass to send off a specific creation message. The message will be stored in the proxy to get the entity handle back later when the server-side graphics entity has been created.

```cpp
void Graphics::GraphicsEntity::OnTransformChanged() [protected, virtual, inherited]
```

called when transform matrix changed

Called by `SetTransform()`. This gives subclasses a chance to react to changes on the transformation matrix.

Reimplemented in `Graphics::CameraEntity`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Graphics::GraphicsEntity Class Reference

#include <graphicsentity.h>

Inheritance diagram for Graphics::GraphicsEntity:
Detailed Description

Client-side proxy of a `InternalGraphics::InternalGraphicsEntity`.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GraphicsEntity()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>virtual ~GraphicsEntity()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>bool IsValid()</code> const</td>
<td>return true if entity is valid (is attached to a stage)</td>
</tr>
<tr>
<td><code>GraphicsEntityType::Code</code> GetType()` const</td>
<td>get the entity type</td>
</tr>
<tr>
<td><code>void SetTransform(const Math::matrix44 &amp;m)</code></td>
<td>set the entity's world space transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetTransform()</code> const</td>
<td>get the entity's world space transform</td>
</tr>
<tr>
<td><code>void SetVisible(bool b)</code></td>
<td>set the entity's visibility</td>
</tr>
<tr>
<td><code>bool IsVisible()</code> const</td>
<td>return true if entity is set to visible</td>
</tr>
<tr>
<td><code>const Ptr&lt; Stage &gt; &amp; GetStage()</code> const</td>
<td>get the stage proxy this entity is attached to</td>
</tr>
<tr>
<td><code>Math::ClipStatus::Type</code> GetClipStatus()` const</td>
<td>get current clip status</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetLocalBoundingBox()</code> const</td>
<td>get the local bounding box</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetGlobalBoundingBox()</code> const</td>
<td>get the global bounding box</td>
</tr>
<tr>
<td><code>Timing::Time GetEntityTime()</code> const</td>
<td>get current entity time</td>
</tr>
<tr>
<td><code>void SendMsg(const Ptr&lt; GraphicsEntityMessage &gt; &amp;msg)</code></td>
<td>send a message to the server-side graphics entity</td>
</tr>
<tr>
<td><code>bool IsObjectRefValid()</code> const</td>
<td>test if the entity's object ref is valid</td>
</tr>
<tr>
<td><code>const Ptr&lt; Threading::ObjectRef &gt; &amp; GetObjectRef()</code> const</td>
<td>test if the entity's object ref is valid</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetRefCount()</code></td>
<td>int GetRefCount () const &lt;br&gt;get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>void AddRef () &lt;br&gt;increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>void Release () &lt;br&gt;decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const &lt;br&gt;return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code></td>
<td>bool IsInstanceOf (const Util::String &amp;className) const &lt;br&gt;return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const &lt;br&gt;return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti)</code></td>
<td>bool IsA (const Rtti &amp;rtti) const &lt;br&gt;return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName)</code></td>
<td>bool IsA (const Util::String &amp;rttiName) const &lt;br&gt;return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const &lt;br&gt;return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName()</code></td>
<td>const Util::String &amp; GetClassName () const &lt;br&gt;get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC()</code></td>
<td>Util::FourCC GetClassFourCC () const &lt;br&gt;get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetType (GraphicsEntityType::Code t)</code></td>
<td>set graphics entity type, called from constructor of subclass</td>
</tr>
<tr>
<td><code>virtual void Setup (const Ptr&lt; Stage &gt; &amp;stage)</code></td>
<td>called by stage when entity should setup itself</td>
</tr>
<tr>
<td><code>virtual void Discard ()</code></td>
<td>called by stage when entity should discard itself</td>
</tr>
<tr>
<td><code>virtual void OnSetupSharedData ()</code></td>
<td>called to setup the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td><code>virtual void OnDiscardSharedData ()</code></td>
<td>called to discard the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td><code>void SendCreateMsg (const Ptr&lt; CreateGraphicsEntity &gt; &amp;msg)</code></td>
<td>send off a specific create message from the subclass</td>
</tr>
<tr>
<td><code>virtual void OnTransformChanged ()</code></td>
<td>called when transform matrix changed</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Graphics::GraphicsEntity::SendMsg (const Ptr<GraphicsEntityMessage> msg &)
```

send a message to the server-side graphics entity

Send a generic GraphicsEntityMessage to the server-side entity.

```cpp
void Graphics::GraphicsEntity::Setup (const Ptr<Stage> &stage_ ) [protected, virtual]
```

called by stage when entity should setup itself

Setup the graphics entity. Subclasses must send a specific creation message in this method. This method is called from StageProxy::AttachEntityProxy().

Reimplemented in `Graphics::CameraEntity`.

```cpp
void Graphics::GraphicsEntity::Discard ( ) [protected, virtual]
```

called by stage when entity should discard itself

Discard the server-side graphics entity. This method is called from StageProxy::RemoveEntityProxy(). There's special handling if the server-side entity hasn't been created yet, in this case, a pointer to the original create message must be handed over to the render thread.

Reimplemented in `Graphics::CameraEntity`.

```cpp
void Graphics::GraphicsEntity::OnSetupSharedData ( ) [protected, virtual]
```

called to setup the FrameSyncSharedData object of the entity

Setup the shared data object. If subclasses wish to add more
members to the shared data object, they must subclass from
InternalGraphics::InternalGraphicsEntitySharedData, setup a
FrameSyncSharedData object with this data type, and NOT CALL the
parent class method! So the lowest subclass defines the type of the
shared data!

```cpp
void Graphics::GraphicsEntity::OnDiscardSharedData() [protected, virtual]
```
called to discard the FrameSyncSharedData object of the entity

Discard the shared data object. See `OnSetupSharedData()` for
details!

```cpp
void Graphics::GraphicsEntity::SendCreateMsg(const Ptr<CreateGraphicsEntity msg> &)
```
send off a specific create message from the subclass

This method must be called from the `Setup()` method of a subclass to
send off a specific creation message. The message will be stored in
the proxy to get the entity handle back later when the server-side
graphics entity has been created.

```cpp
void Graphics::GraphicsEntity::OnTransformChanged() [protected, virtual]
```
called when transform matrix changed

Called by `SetTransform()`. This gives subclasses a chance to react to
changes on the transformation matrix.

Reimplemented in `Graphics::CameraEntity`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Graphics::GraphicsEntityType
Graphics::GraphicsEntityType Class Reference

#include <graphicsentitytype.h>
Detailed Description

Define graphics entity types. Must be identical with InternalGraphics::InternalGrapicsEntityType.

(C) 2008 Radon Labs GmbH
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**Graphics::GraphicsHandler**
Graphics::GraphicsHandler Class Reference

#include <graphicshandler.h>

Inheritance diagram for Graphics::GraphicsHandler:
Detailed Description

Runs in the graphics thread context, setup the graphics runtime environment and processes messages to the graphics thread.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GraphicsHandler()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~GraphicsHandler()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>Open the handler</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>Close the handler</td>
</tr>
<tr>
<td><code>HandleMessage(const Messaging::Message &amp;msg)</code></td>
<td>Handle a message, return true if handled</td>
</tr>
<tr>
<td><code>DoWork()</code></td>
<td>Do per-frame work</td>
</tr>
<tr>
<td><code>WaitForPendingResources()</code></td>
<td>Wait for pending resources, this method is called in the main thread context!</td>
</tr>
<tr>
<td><code>SetCompanyName(const Util::StringAtom &amp;companyName)</code></td>
<td>Set the company name</td>
</tr>
<tr>
<td><code>GetCompanyName()</code> const</td>
<td>Get the company name</td>
</tr>
<tr>
<td><code>SetAppName(const Util::StringAtom &amp;appName)</code></td>
<td>Set the application name</td>
</tr>
<tr>
<td><code>GetAppName()</code> const</td>
<td>Get the application name</td>
</tr>
<tr>
<td><code>IsOpen()</code></td>
<td>Return true if open</td>
</tr>
<tr>
<td><code>GetRefCount()</code> const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Rtti &amp;rtti)</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;className)</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC)</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti)</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName)</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC)</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName ()</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC ()</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only)!</td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Graphics::GraphicsHandler::DoWork ( ) [virtual]
do per-frame work

This is the per-frame method which implements the asynchronous render-loop.

Reimplemented from Interface::InterfaceHandlerBase.

void
Graphics::GraphicsHandler::WaitForPendingResources ( )
wait for pending resources, this method is called in the main thread context!

Waits for all resources to be loaded. THIS METHOD IS CALLED IN THE MAIN THREAD CONTEXT!

int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]
get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]
increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

```c++
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```c++
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```c++
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Graphics::GraphicsInterface
#include <graphicsinterface.h>

Inheritance diagram for Graphics::GraphicsInterface:
Detailed Description

Implements the asynchronous interface to the InternalGraphics subsystem. Usually the application doesn’t call methods on the GraphicsInterface directly, but instead uses one of the Graphics objects to communicate with the graphics thread.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GraphicsInterface ()</strong></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td><strong>virtual ~GraphicsInterface ()</strong></td>
<td><em>destructor</em></td>
</tr>
<tr>
<td><strong>virtual void Open ()</strong></td>
<td><em>open the interface object</em></td>
</tr>
<tr>
<td><strong>virtual void Close ()</strong></td>
<td><em>close the interface object</em></td>
</tr>
<tr>
<td><strong>void SendBatched (const Ptr &lt; Messaging::Message &gt; &amp;msg)</strong></td>
<td><em>send a batched messages (call FlushBatchedMessages() to send them to render thread)</em></td>
</tr>
<tr>
<td><strong>void FlushBatchedMessages ()</strong></td>
<td><em>flush batched messages, call before WaitForFrameEvent()</em></td>
</tr>
<tr>
<td><strong>void WaitForPendingResources ()</strong></td>
<td><em>wait until all pending resources are loaded</em></td>
</tr>
<tr>
<td><strong>void EnterLockStepMode ()</strong></td>
<td><em>enter lock-step mode</em></td>
</tr>
<tr>
<td><strong>void LeaveLockStepMode ()</strong></td>
<td><em>leave lock-step mode</em></td>
</tr>
<tr>
<td><strong>void GameThreadWaitForFrameSync ()</strong></td>
<td><em>call when game thread arrives at frame sync point</em></td>
</tr>
<tr>
<td><strong>virtual void AttachHandler (const Ptr &lt; Messaging::Handler &gt; &amp;h)</strong></td>
<td><em>attach a handler to the port (call before open!)</em></td>
</tr>
<tr>
<td><strong>const Util::StringAtom &amp; GetCompanyName () const</strong></td>
<td><em>get the company name</em></td>
</tr>
<tr>
<td><strong>const Util::StringAtom &amp; GetAppName () const</strong></td>
<td><em>get the application name</em></td>
</tr>
<tr>
<td><strong>const Util::StringAtom &amp; GetRootDirectory () const</strong></td>
<td><em>get the root directory</em></td>
</tr>
<tr>
<td><strong>SetHandlerThread (const Ptr &lt;</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>HandlerThreadBase &amp;handlerThread</code></td>
<td>set pointer to handler thread object (must be derived from <code>HandlerThreadBase</code>)</td>
</tr>
<tr>
<td><code>GetHandlerThread()</code> const</td>
<td>get pointer to handler thread object</td>
</tr>
<tr>
<td><code>RemoveHandler(const Ptr&lt; Handler &gt;&amp;h)</code></td>
<td>dynamically remove a handler from the port</td>
</tr>
<tr>
<td><code>IsOpen()</code> const</td>
<td>return true if port is open</td>
</tr>
<tr>
<td><code>Send(const Ptr&lt; MESSAGETYPE &gt;&amp;msg)</code></td>
<td>send an asynchronous message to the port</td>
</tr>
<tr>
<td><code>SendWait(const Ptr&lt; MESSAGETYPE &gt;&amp;msg)</code></td>
<td>send a message and wait for completion</td>
</tr>
<tr>
<td><code>Wait(const Ptr&lt; MESSAGETYPE &gt;&amp;msg)</code></td>
<td>wait for a message to be handled</td>
</tr>
<tr>
<td><code>Peek(const Ptr&lt; MESSAGETYPE &gt;&amp;msg)</code></td>
<td>peek a message whether it has been handled</td>
</tr>
<tr>
<td><code>Cancel(const Ptr&lt; MESSAGETYPE &gt;&amp;msg)</code></td>
<td>cancel a pending message</td>
</tr>
<tr>
<td><code>GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti&amp;)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String&amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC)</code></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>&amp;classFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

```c++
void SignalPendingResourceEvent ()
  signal the resource loaded event, called by GraphicsHandler
```
Member Function Documentation

```cpp
void Graphics::GraphicsInterface::SendBatched
    (const Ptr<Messaging::Message> &msg)
send a batched messages (call FlushBatchedMessages() to send them to render thread)

Send a batched message. The message will not be sent to the render thread immediately, but instead will be collected in a batch message for later sending. Use this method to reduce communication overhead between the main and render thread by sending only one batch message instead of many messages. This method may only be called from the main thread!

```cpp
void Graphics::GraphicsInterface::FlushBatchedMessages()
flush batched messages, call before WaitForFrameEvent()

Send off the batched messages to the render thread. This reduces thread synchronization overhead dramatically if many messages must be sent per-frame. Only the main thread may call this method.

```cpp
void Graphics::GraphicsInterface::WaitForPendingResources()
wait until all pending resources are loaded

This method will wait until the graphics handler signals that all pending resources have been loaded.

```cpp
void Messaging::AsyncPort::RemoveHandler
    (const Ptr<Handler> &h)
dynamically remove a handler from the port
Dynamically remove a message handler.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Graphics::GraphicsServer
Graphics::GraphicsServer Class Reference

#include <graphicsserver.h>

Inheritance diagram for Graphics::GraphicsServer:
Detailed Description

Client-side proxy of the InternalGraphicsServer. Used to create and update Stages and Views.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GraphicsServer ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>virtual ~GraphicsServer ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>void Open ()</code></td>
<td>Open the graphics server</td>
</tr>
<tr>
<td><code>void Close ()</code></td>
<td>Close the graphics server</td>
</tr>
<tr>
<td><code>bool IsOpen () const</code></td>
<td>Return true if graphics server is open</td>
</tr>
<tr>
<td><code>void OnFrame ()</code></td>
<td>Perform client-side per-frame updates</td>
</tr>
<tr>
<td><code>Ptr&lt; Stage &gt; CreateStage (const Util::StringAtom &amp;name, Util::Array&lt; Ptr&lt; Visibility::VisibilitySystemBase &gt; &gt; &amp;visSystems)</code></td>
<td>Create a stage</td>
</tr>
<tr>
<td><code>void DiscardStage (const Ptr&lt; Stage &gt; &amp;stage)</code></td>
<td>Discard a stage object</td>
</tr>
<tr>
<td><code>void DiscardAllStages ()</code></td>
<td>Discard all stage objects</td>
</tr>
<tr>
<td><code>bool HasStage (const Util::StringAtom &amp;name)</code></td>
<td>Return true if a stage exists by name</td>
</tr>
<tr>
<td><code>const Ptr&lt; Stage &gt; &amp; GetStageByName (const Util::StringAtom &amp;name)</code></td>
<td>Lookup a stage by name</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Ptr&lt; Stage &gt; &gt; &amp; GetStages () const</code></td>
<td>Get all stages</td>
</tr>
<tr>
<td><code>CreateView (const Core::Rtti &amp;viewClass, const</code></td>
<td>Create view</td>
</tr>
</tbody>
</table>
Ptr<View> create a view

void DiscardView (const Ptr&view) discard a view

void DiscardAllViews () discard all views

bool HasView (const Util::StringAtom &name) const return true if a view exists by name

const Ptr<View> & GetViewByViewName (const Util::StringAtom &name) lookup a view by name

const Util::Array<Ptr<View>> & GetViews () const get all view

void SetDefaultView (const Ptr&view) set the default view

const Ptr<View> & GetDefaultView () const get the default view

uint GetFrameCount () const get frame count

void RegisterRenderModule (Ptr<RenderModules::RenderModule> &renderModule) register a render module

void UnregisterRenderModule (Ptr<RenderModules::RenderModule> &renderModule) unregister a render module
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const Util::Array&lt; Ptr&lt; RenderModules::RenderModule &gt; &gt; &amp; GetRenderModules()</code></td>
<td>get array of all currently registered render modules</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName()</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC()</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```c++
void Graphics::GraphicsServer::DiscardStage(const Ptr<Stage> &stage)

discard a stage object

FIXME FIXME FIXME: this method is broken when more then 1 stage exists (indices in stageIndexMap are broken after first stage is removed).

void Graphics::GraphicsServer::DiscardView(const Ptr<View> &view)

discard a view

FIXME FIXME FIXME: this method is broken when more then 1 view exists (indices in viewIndexMap are broken after first view is removed).

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

\begin{verbatim}
const Util::String & Core::RefCounted::GetClassName () const [inline, inherited]
\end{verbatim}

get the class name

Get the class name of the object.

\begin{verbatim}
Util::FourCC Core::RefCounted::GetClassFourCC () const [inline, inherited]
\end{verbatim}

get the class FourCC code

Get the class FourCC of the object.

\begin{verbatim}
void Core::RefCounted::DumpRefCountingLeaks () [static, inherited]
\end{verbatim}

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Graphics::ModelEntity
Graphics::ModelEntity Class Reference

#include <modelentity.h>

Inheritance diagram for Graphics::ModelEntity:
Detailed Description

Client-side proxy of a `InternalGraphics::InternalModelEntity`.

(C) 2008 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ModelEntity ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~ModelEntity ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void SetResourceId(const Resources::ResourceId &amp;resId)</strong></td>
<td>set the model's resource id</td>
</tr>
<tr>
<td><strong>const Resources::ResourceId &amp; GetResourceId () const</strong></td>
<td>get the model's resource id</td>
</tr>
<tr>
<td><strong>void SetRootNodePath(const Util::StringAtom &amp;rootNodePath)</strong></td>
<td>set optional root node path (allows to setup from a child node in a model node hierarchy)</td>
</tr>
<tr>
<td><strong>const Util::StringAtom &amp; GetRootNodePath () const</strong></td>
<td>get optional root node path</td>
</tr>
<tr>
<td><strong>void SetRootNodeOffsetMatrix(const Math::matrix44 &amp;offsetMatrix)</strong></td>
<td>set optional root node offset matrix</td>
</tr>
<tr>
<td><strong>const Math::matrix44 &amp; GetRootNodeOffsetMatrix () const</strong></td>
<td>get optional root node offset matrix</td>
</tr>
<tr>
<td><strong>void ConfigureAnimDrivenMotionTracking(bool enabled, const Util::StringAtom &amp;jointName)</strong></td>
<td>enable anim driven motion tracking</td>
</tr>
<tr>
<td><strong>void ConfigureAnimEventTracking(bool enabled, bool onlyDominatingClip)</strong></td>
<td>enable anim event tracking</td>
</tr>
<tr>
<td><strong>void ConfigureCharJointTracking(bool enabled, const Util::Array<a href="">Util::StringAtom</a> &amp;jointNames)</strong></td>
<td>enable joint tracking</td>
</tr>
<tr>
<td><strong>bool IsAnimDrivenMotionTrackingEnabled() const</strong></td>
<td>return true if anim driven motion tracking is enabled</td>
</tr>
<tr>
<td>C++ Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetAnimDrivenMotionJointName()</code></td>
<td>get the anim-driven-motion tracking joint name</td>
</tr>
<tr>
<td><code>const Math::vector &amp; GetAnimDrivenMotionVector()</code></td>
<td>get the current anim-driven-motion tracking vector</td>
</tr>
<tr>
<td><code>bool IsAnimEventTrackingEnabled()</code></td>
<td>return true if anim event tracking is enabled</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Animation::AnimEventInfo &gt; &amp; GetAnimEvents () const</code></td>
<td>return the array of anim events of the current frame</td>
</tr>
<tr>
<td><code>bool IsCharJointTrackingEnabled () const</code></td>
<td>return true if char joint tracking is enabled</td>
</tr>
<tr>
<td><code>bool IsCharJointDataValid () const</code></td>
<td>return true if current char joint data is valid                         (may change by visibility)</td>
</tr>
<tr>
<td><code>void AddTrackedCharJoint (const Util::StringAtom &amp;jointName)</code></td>
<td>dynamically add a tracked character joint</td>
</tr>
<tr>
<td><code>const Shared::CharJointInfo * GetTrackedCharJointInfo (const Util::StringAtom &amp;jointName) const</code></td>
<td>access tracked char joint data, return NULL ptr if joint data not valid or not yet available!</td>
</tr>
<tr>
<td><code>void UpdateModelNodeInstanceShaderVariable (const Util::StringAtom &amp;nodeName, Util::StringAtom &amp;variableSemantic, Util::Variant &amp;value)</code></td>
<td>update the value of a shader variable on a node instance of the model entity</td>
</tr>
<tr>
<td><code>void UpdateModelNodeInstanceVisibility (Util::StringAtom &amp;nodeName, bool visible)</code></td>
<td>update the visibility of a model node instance</td>
</tr>
<tr>
<td><code>bool IsValid () const</code></td>
<td>return true if entity is valid (is attached to a stage)</td>
</tr>
<tr>
<td><code>GraphicsEntityType::Code GetType () const</code></td>
<td>get the entity type</td>
</tr>
<tr>
<td><code>void SetTransform (const Math::matrix44 &amp;)</code></td>
<td>set the entity's world space transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetTransform () const</code></td>
<td>get the entity's world space transform</td>
</tr>
<tr>
<td><code>void SetVisible (bool b)</code></td>
<td>set the entity's world space transform</td>
</tr>
</tbody>
</table>
bool IsVisible () const
return true if entity is set to visible

const Ptr< Stage > & GetStage () const
get the stage proxy this entity is attached to

Math::ClipStatus::Type GetClipStatus () const
get current clip status

const Math::bbox & GetLocalBoundingBox () const
get the local bounding box

const Math::bbox & GetGlobalBoundingBox () const
get the global bounding box

Timing::Time GetEntityTime () const
get current entity time

void SendMsg (const Ptr< GraphicsEntityMessage > &msg)
send a message to the server-side graphics

bool IsObjectRefValid () const
test if the entity's object ref is valid

const Ptr< Threading::ObjectRef > & GetObjectRef () const
get the entity handle

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given derived class
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool IsA (const Util::String &amp;rttiName)</td>
<td>return true if this object is instance of given derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC)</td>
<td>return true if this object is instance of given derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC        GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
</table>

_dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)_
## Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetType</strong> <em>(GraphicsEntityType::Code t)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set graphics entity type, called from constructor of subclass</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>virtual void</th>
<th><strong>Discard</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>called by stage when entity should discard itself</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SendCreateMsg</strong> <em>(const Ptr&lt; CreateGraphicsEntity &gt; &amp;msg)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>send off a specific create message from the subclass</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>virtual void</th>
<th><strong>OnTransformChanged</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>called when transform matrix changed</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```
const Shared::CharJointInfo *
Graphics::ModelEntity::GetTrackedCharJointInfo
  ( const Util::StringAtom jointName ) const
```

access tracked char joint data, return NULL ptr if joint data not valid or not yet available!

**NOTE:** the method returns 0 if the joint is not valid or has not been found!

```
void
Graphics::GraphicsEntity::SendMsg
  ( const Ptr< GraphicsEntityMessage msg > ) [inherited]
```

send a message to the server-side graphics entity

Send a generic GraphicsEntityMessage to the server-side entity.

```
void
Graphics::GraphicsEntity::Discard ( ) [protected, virtual, inherited]
```

called by stage when entity should discard itself

Discard the server-side graphics entity. This method is called from StageProxy::RemoveEntityProxy(). There's special handling if the server-side entity hasn't been created yet, in this case, a pointer to the original create message must be handed over to the render thread.

Reimplemented in Graphics::CameraEntity.

```
void
Graphics::GraphicsEntity::SendCreateMsg
  ( const Ptr< CreateGraphicsEntity msg > ) [protected, inherited]
```

send off a specific create message from the subclass

This method must be called from the **Setup()** method of a subclass to send off a specific creation message. The message will be stored in the proxy to get the entity handle back later when the server-side
graphics entity has been created.

```cpp
void Graphics::GraphicsEntity::OnTransformChanged() [protected, virtual, inherited]
```
called when transform matrix changed

Called by `SetTransform()`. This gives subclasses a chance to react to changes on the transformation matrix.

Reimplemented in `Graphics::CameraEntity`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```c
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Graphics::MouseRenderer
Graphics::MouseRenderer Class Reference

#include <mouserenderer.h>

Inheritance diagram for Graphics::MouseRenderer:
Detailed Description

Main-thread proxy for the MouseRenderDevice (renders mouse pointers).

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MouseRenderer</strong> () virtual ~MouseRenderer()</td>
<td>Constructor and destructor</td>
</tr>
<tr>
<td><strong>PreloadTextures</strong> (const Util::Array/Resources::ResourceId &amp;resIds)</td>
<td>Pre-load mouse pointer textures</td>
</tr>
<tr>
<td><strong>UpdatePointers</strong> (const Util::Array&lt;MousePointer &amp;pointers)</td>
<td>Update mouse pointers</td>
</tr>
<tr>
<td><strong>GetRefCount</strong> () const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef</strong> ()</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><strong>Release</strong> ()</td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><strong>GetClassName</strong> () const</td>
<td>Get the class name</td>
</tr>
<tr>
<td>Utility::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```c++
int Core::RefCounted::GetRefCount( ) const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```c++
void Core::RefCounted::AddRef( ) [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```c++
void Core::RefCounted::Release( ) [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```c++
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
```
get the class name

Get the class name of the object.

```c++
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```c++
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Graphics::PointLightEntity
Graphics::PointLightEntity Class Reference

#include <pointlightentity.h>

Inheritance diagram for Graphics::PointLightEntity:
Detailed Description

Client-side proxy of a InternalGraphics::InternalPointLightEntity.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PointLightEntity</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>SetTransformFromPosDirAndRange</strong> (const Math::point &amp;pos, const Math::vector &amp;dir, float range)</td>
<td>Set transform from position and range</td>
</tr>
<tr>
<td>Lighting::LightType::Code <strong>GetLightType</strong> () const</td>
<td>Get the light type (defined by subclass)</td>
</tr>
<tr>
<td>void <strong>SetColor</strong> (const Math::float4 &amp;c)</td>
<td>Set primary light color</td>
</tr>
<tr>
<td>const Math::float4 &amp; <strong>GetColor</strong> () const</td>
<td>Get primary light color</td>
</tr>
<tr>
<td>void <strong>SetCastShadows</strong> (bool b)</td>
<td>Enable/disable shadow casting</td>
</tr>
<tr>
<td>bool <strong>GetCastShadows</strong> () const</td>
<td>Get shadow casting flag</td>
</tr>
<tr>
<td>void <strong>SetProjMapUvOffsetAndScale</strong> (const Math::float4 &amp;v)</td>
<td>Set projection map UV offset and scale (xy-&gt;offset, zw-&gt;scale)</td>
</tr>
<tr>
<td>const Math::float4 &amp; <strong>GetProjMapUvOffsetAndScale</strong> () const</td>
<td>Get projection map UV offset and scale</td>
</tr>
<tr>
<td>bool <strong>IsValid</strong> () const</td>
<td>Return true if entity is valid (is attached to a stage)</td>
</tr>
<tr>
<td>GraphicsEntityType::Code <strong>GetType</strong> () const</td>
<td>Get the entity type</td>
</tr>
<tr>
<td>void <strong>SetTransform</strong> (const Math::matrix44 &amp;m)</td>
<td>Set the entity’s world space transform</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; <strong>GetTransform</strong> () const</td>
<td>Get the entity’s world space transform</td>
</tr>
<tr>
<td>void <strong>SetVisible</strong> (bool b)</td>
<td>Set the entity’s visibility</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>bool</strong> IsVisible () const</td>
<td>return true if entity is set to visible</td>
</tr>
<tr>
<td>const <strong>Ptr</strong>&lt; Stage &gt; &amp; GetStage () const</td>
<td>get the stage proxy this entity is attached to</td>
</tr>
<tr>
<td><strong>Math::ClipStatus::Type</strong> GetClipStatus () const</td>
<td>get current clip status</td>
</tr>
<tr>
<td>const <strong>Math::bbox</strong> &amp; GetLocalBoundingBox () const</td>
<td>get the local bounding box</td>
</tr>
<tr>
<td>const <strong>Math::bbox</strong> &amp; GetGlobalBoundingBox () const</td>
<td>get the global bounding box</td>
</tr>
<tr>
<td><strong>Timing::Time</strong> GetEntityTime () const</td>
<td>get current entity time</td>
</tr>
<tr>
<td><strong>void</strong> SendMsg (const <strong>Ptr</strong>&lt; GraphicsEntityMessage &gt; &amp;msg)</td>
<td>send a message to the server-side graphics entity</td>
</tr>
<tr>
<td><strong>bool</strong> IsObjectRefValid () const</td>
<td>test if the entity's object ref is valid</td>
</tr>
<tr>
<td>const <strong>Ptr</strong>&lt; Threading::ObjectRef &gt; &amp; GetObjectRef () const</td>
<td>get the entity handle</td>
</tr>
<tr>
<td><strong>int</strong> GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void</strong> AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void</strong> Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool</strong> IsInstanceOf (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool</strong> IsInstanceOf (const <strong>Util::String</strong> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool</strong> IsInstanceOf (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>
| **bool** IsA (const **Rtti** &rtti) const | return true if this object is instance of given class,
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td><code>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</code></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void SetLightType (Lighting::LightType::Code c)</td>
<td>set the light type (must be called from sub-classes constructor</td>
</tr>
<tr>
<td>void SetType (GraphicsEntityType::Code t)</td>
<td>set graphics entity type, called from constructor of subclass</td>
</tr>
<tr>
<td>virtual void Discard ()</td>
<td>called by stage when entity should discard itself</td>
</tr>
<tr>
<td>virtual void OnSetupSharedData ()</td>
<td>called to setup the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td>virtual void OnDiscardSharedData ()</td>
<td>called to discard the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td>void SendCreateMsg (const Ptr&lt; CreateGraphicsEntity &gt; &amp;msg)</td>
<td>send off a specific create message from the subclass</td>
</tr>
<tr>
<td>virtual void OnTransformChanged ()</td>
<td>called when transform matrix changed</td>
</tr>
</tbody>
</table>
### Member Function Documentation

```cpp
template<typename T>
void Graphics::GraphicsEntity::SendMsg (const Ptr<GraphicsEntityMessage> msg ) [inherited]
```

send a message to the server-side graphics entity

Send a generic GraphicsEntityMessage to the server-side entity.

```cpp
template<typename T>
void Graphics::GraphicsEntity::Discard ( ) [protected, virtual, inherited]
```

called by stage when entity should discard itself

Discard the server-side graphics entity. This method is called from StageProxy::RemoveEntityProxy(). There's special handling if the server-side entity hasn't been created yet, in this case, a pointer to the original create message must be handed over to the render thread.

Reimplemented in `Graphics::CameraEntity`.

```cpp
template<typename T>
void Graphics::GraphicsEntity::OnSetupSharedData ( ) [protected, virtual, inherited]
```

called to setup the FrameSyncSharedData object of the entity

Setup the shared data object. If subclasses wish to add more members to the shared data object, they must subclass from InternalGraphics::InternalGraphicsEntitySharedData, setup a FrameSyncSharedData object with this data type, and NOT CALL the parent class method! So the lowest subclass defines the type of the shared data!

```cpp
template<typename T>
void Graphics::GraphicsEntity::OnDiscardSharedData ( ) [protected, virtual, inherited]
```

called to discard the FrameSyncSharedData object of the entity

Discard the shared data object. See `OnSetupSharedData()` for
void Graphics::GraphicsEntity::SendCreateMsg (const Ptr<CreateGraphicsEntity> msg) [protected, inherited]

send off a specific create message from the subclass

This method must be called from the Setup() method of a subclass to send off a specific creation message. The message will be stored in the proxy to get the entity handle back later when the server-side graphics entity has been created.

void Graphics::GraphicsEntity::OnTransformChanged ( ) [protected, virtual, inherited]
called when transform matrix changed

Called by SetTransform(). This gives subclasses a chance to react to changes on the transformation matrix.

Reimplemented in Graphics::CameraEntity.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]
imcrement refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Graphics::SpotLightEntity
Graphics::SpotLightEntity Class Reference

#include <spotlightentity.h>

Inheritance diagram for Graphics::SpotLightEntity:
Detailed Description

Client-side proxy of a InternalGraphics::InternalSpotLightEntity.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SpotLightEntity ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>SetTransformFromPosDirRangeAndCone (const Math::point &amp;pos, const Math::vector &amp;dir, float range, float coneAngle)</strong></td>
<td>Set transform from pos, dir, range and cone</td>
</tr>
<tr>
<td><strong>GetLightType () const</strong></td>
<td>Get the light type (defined by subclass)</td>
</tr>
<tr>
<td><strong>SetColor (const Math::float4 &amp;c)</strong></td>
<td>Set primary light color</td>
</tr>
<tr>
<td><strong>GetColor () const</strong></td>
<td>Get primary light color</td>
</tr>
<tr>
<td><strong>SetCastShadows (bool b)</strong></td>
<td>Enable/disable shadow casting</td>
</tr>
<tr>
<td><strong>GetCastShadows () const</strong></td>
<td>Get shadow casting flag</td>
</tr>
<tr>
<td><strong>SetProjMapUvOffsetAndScale (const Math::float4 &amp;v)</strong></td>
<td>Set projection map UV offset and scale (xy-&gt;offset, zw-&gt;scale)</td>
</tr>
<tr>
<td><strong>GetProjMapUvOffsetAndScale () const</strong></td>
<td>Get projection map UV offset and scale</td>
</tr>
<tr>
<td><strong>IsValid () const</strong></td>
<td>Return true if entity is valid (is attached to a stage)</td>
</tr>
<tr>
<td><strong>GetType () const</strong></td>
<td>Get the entity type</td>
</tr>
<tr>
<td><strong>SetTransform (const Math::matrix44 &amp;m)</strong></td>
<td>Set the entity's world space transform</td>
</tr>
<tr>
<td><strong>GetTransform () const</strong></td>
<td>Get the entity's world space transform</td>
</tr>
<tr>
<td><strong>SetVisible (bool b)</strong></td>
<td>Set the entity's visibility</td>
</tr>
<tr>
<td><strong>bool</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>IsVisible</td>
<td>const\nreturn true if entity is set to visible</td>
</tr>
<tr>
<td>GetStage</td>
<td>const\nget the stage proxy this entity is attached to</td>
</tr>
<tr>
<td>GetClipStatus</td>
<td>const\nget current clip status</td>
</tr>
<tr>
<td>GetLocalBoundingBox</td>
<td>const\nget the local bounding box</td>
</tr>
<tr>
<td>GetGlobalBoundingBox</td>
<td>const\nget the global bounding box</td>
</tr>
<tr>
<td>GetEntityTime</td>
<td>const\nget current entity time</td>
</tr>
<tr>
<td>SendMsg</td>
<td>(const Ptr&lt;GraphicsEntityMessage&gt; &amp;msg)\nsend a message to the server-side graphics entity</td>
</tr>
<tr>
<td>IsObjectRefValid</td>
<td>const\ntest if the entity's object ref is valid</td>
</tr>
<tr>
<td>GetObjectRef</td>
<td>const\nget the entity handle</td>
</tr>
<tr>
<td>GetRefCount</td>
<td>const\nget the current refcount</td>
</tr>
<tr>
<td>AddRef</td>
<td>()\nincrement refcount by one</td>
</tr>
<tr>
<td>Release</td>
<td>()\ndecrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>IsInstanceOf</td>
<td>(const Rtti &amp;rtti)\nreturn true if this object is instance of given class</td>
</tr>
<tr>
<td>IsInstanceOf</td>
<td>(const Util::String &amp;className)\nreturn true if this object is instance of given class by string</td>
</tr>
<tr>
<td>IsInstanceOf</td>
<td>(const Util::FourCC &amp;classFourCC)\nreturn true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>IsA</td>
<td>(const Rtti &amp;rtti)\nreturn true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>IsA</td>
<td>(const Util::String &amp;rttiName)\nreturn true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <em>Util::FourCC</em> &amp;rttiFourCC) const</td>
<td></td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
<tr>
<td>const <em>Util::String</em> &amp; <strong>GetClassName</strong> () const</td>
<td></td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td><em>Util::FourCC</em> <strong>GetClassFourCC</strong> () const</td>
<td></td>
</tr>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetLightType (Lighting::LightType::Code c)</code></td>
<td>set the light type (must be called from sub-classes constructor)</td>
</tr>
<tr>
<td><code>void SetType (GraphicsEntityType::Code t)</code></td>
<td>set graphics entity type, called from constructor of subclass</td>
</tr>
<tr>
<td><code>virtual void Discard ()</code></td>
<td>called by stage when entity should discard itself</td>
</tr>
<tr>
<td><code>virtual void OnSetupSharedData ()</code></td>
<td>called to setup the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td><code>virtual void OnDiscardSharedData ()</code></td>
<td>called to discard the FrameSyncSharedData object of the entity</td>
</tr>
<tr>
<td><code>void SendCreateMsg (const Ptr&lt; CreateGraphicsEntity &gt;&amp; msg)</code></td>
<td>send off a specific create message from the subclass</td>
</tr>
<tr>
<td><code>virtual void OnTransformChanged ()</code></td>
<td>called when transform matrix changed</td>
</tr>
</tbody>
</table>
void Graphics::GraphicsEntity::SendMsg ( const Ptr<GraphicsEntityMessage> msg ) [inherited]

Send a generic GraphicsEntityMessage to the server-side entity.

called by stage when entity should discard itself

Discard the server-side graphics entity. This method is called from StageProxy::RemoveEntityProxy(). There's special handling if the server-side entity hasn't been created yet, in this case, a pointer to the original create message must be handed over to the render thread.

Reimplemented in Graphics::CameraEntity.

void Graphics::GraphicsEntity::OnSetupSharedData ( ) [protected, virtual, inherited]

called to setup the FrameSyncSharedData object of the entity

Setup the shared data object. If subclasses wish to add more members to the shared data object, they must subclass from InternalGraphics::InternalGraphicsEntitySharedData, setup a FrameSyncSharedData object with this data type, and NOT CALL the parent class method! So the lowest subclass defines the type of the shared data!

void Graphics::GraphicsEntity::OnDiscardSharedData ( ) [protected, virtual, inherited]

called to discard the FrameSyncSharedData object of the entity

Discard the shared data object. See OnSetupSharedData() for
details!

```cpp
void Graphics::GraphicsEntity::SendCreateMsg(const Ptr<CreateGraphicsEntity> &msg) [protected, inherited]
```

send off a specific create message from the subclass

This method must be called from the **Setup()** method of a subclass to send off a specific creation message. The message will be stored in the proxy to get the entity handle back later when the server-side graphics entity has been created.

```cpp
void Graphics::GraphicsEntity::OnTransformChanged() [protected, virtual, inherited]
```

called when transform matrix changed

Called by **SetTransform()**. This gives subclasses a chance to react to changes on the transformation matrix.

Reimplemented in **Graphics::CameraEntity**.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrease refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

```
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Graphics::View
#include <view.h>

Inheritance diagram for Graphics::View:
Detailed Description

A client-side proxy of a `InternalGraphics::InternalView` in the Graphic subsystem.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Member Functions</td>
<td></td>
</tr>
<tr>
<td><strong>View ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~View ()</td>
<td>destructor</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>return true if the View is valid</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; GetName () const</td>
<td>get name of view</td>
</tr>
<tr>
<td>const Core::Rtti &amp; GetViewClass () const</td>
<td>get the class-type of the server-side view object</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; GetStageName () const</td>
<td>get the name of the stage this view is attached to</td>
</tr>
<tr>
<td>const Resources::ResourceId &amp; GetFrameShaderName () const</td>
<td>get the name of the frame-shader this view will use for rendering</td>
</tr>
<tr>
<td>bool IsDefaultView () const</td>
<td>check whether this is the default view</td>
</tr>
<tr>
<td>void SetCameraEntity (const Ptr&lt; CameraEntity &gt; &amp;cameraEntity)</td>
<td>set the camera entity this view should &quot;look through&quot;</td>
</tr>
<tr>
<td>const Ptr&lt; CameraEntity &gt; &amp; GetCameraEntity () const</td>
<td>get the view's current camera entity (may return invalid ptr if no camera is set)</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
</table>

`dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)`
Member Function Documentation

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
GraphicsFeature::ActorGraphicsProperty
GraphicsFeature::ActorGraphicsProperty
Class Reference

#include <actorgraphicsproperty.h>

Inheritance diagram for GraphicsFeature::ActorGraphicsProperty:

```
Core::RefCounted
  |
  v
Messaging::Port
  |
  v
Game::Property
  |
  v
GraphicsFeature::GraphicsProperty
  |
  v
GraphicsFeature::ActorGraphicsProperty
```
Detailed Description

**Graphics** property for animated characters.

(C) 2008 Radon Labs GmbH
Public Types

enum CallbackType

callback types
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void \textbf{SetupDefaultAttributes} ()</td>
<td>setup default entity attributes</td>
</tr>
<tr>
<td>virtual void \textbf{SetupAcceptedMessages} ()</td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>virtual void \textbf{HandleMessage} (const \textbf{Ptr\textless Messaging::Message \textgreater} &amp;msg)</td>
<td>handle a single message</td>
</tr>
<tr>
<td>virtual void \textbf{OnActivate} ()</td>
<td>called from Entity::ActivateProperties()</td>
</tr>
<tr>
<td>virtual void \textbf{OnDeactivate} ()</td>
<td>called from Entity::DeactivateProperties()</td>
</tr>
<tr>
<td>virtual void \textbf{SetupCallbacks} ()</td>
<td>setup callbacks for this property, call by entity in \textbf{OnActivate}()</td>
</tr>
<tr>
<td>virtual const \textbf{Util::String} &amp; \textbf{GetGraphicsResource} () const</td>
<td>override to provide a self managed graphics resource (default is Attr::Graphics)</td>
</tr>
<tr>
<td>virtual void \textbf{OnRenderDebug} ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>const \textbf{Ptr\textless Entity \textgreater} &amp; \textbf{GetEntity} () const</td>
<td>get entity this property is attached to</td>
</tr>
<tr>
<td>bool \textbf{HasEntity} () const</td>
<td>return true if entity pointer is valid</td>
</tr>
<tr>
<td>bool \textbf{IsActive} () const</td>
<td>return true if property is currently active</td>
</tr>
<tr>
<td>virtual void \textbf{OnLoad} ()</td>
<td>called from within Entity::Load() after attributes are loaded</td>
</tr>
<tr>
<td>virtual void \textbf{OnStart} ()</td>
<td>called from within Entity::OnStart() after OnLoad when the complete world exist</td>
</tr>
<tr>
<td>virtual void \textbf{OnSave} ()</td>
<td>called from within Entity::Save() before attributes are saved back to database</td>
</tr>
<tr>
<td>virtual void \textbf{OnBeginFrame} ()</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void OnMoveBefore()</td>
<td>called on begin of frame</td>
</tr>
<tr>
<td>virtual void OnMoveAfter()</td>
<td>called before movement happens</td>
</tr>
<tr>
<td>virtual void OnRender()</td>
<td>called after movement has happened</td>
</tr>
<tr>
<td>virtual void OnLoseActivity()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void OnGainActivity()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>void AttachHandler(const Ptr&lt;Handler&gt; &amp;h)</td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td>void RemoveHandler(const Ptr&lt;Handler&gt; &amp;h)</td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td>void RemoveAllHandlers()</td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td>SizeT GetNumHandlers() const</td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td>const Ptr&lt;Handler&gt; &amp; GetHandlerAtIndex(IndexT i) const</td>
<td>get a message handler by index</td>
</tr>
<tr>
<td>virtual void Send(const Ptr&lt;Message&gt; &amp;msg)</td>
<td>send a message to the port</td>
</tr>
<tr>
<td>const Util::Array&lt;const Id *&gt; &amp; GetAcceptedMessages() const</td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td>bool AcceptsMessage(const Id &amp;msgId) const</td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
</tbody>
</table>
bool *IsInstanceOf* (const Util::String &className) const

return true if this object is instance of given class by string

bool *IsInstanceOf* (const Util::FourCC &classFourCC) const

return true if this object is instance of given class by fourcc

bool *IsA* (const Rtti &rtti) const

return true if this object is instance of given class, or a derived class

bool *IsA* (const Util::String &rttiName) const

return true if this object is instance of given class, or a derived class, by string

bool *IsA* (const Util::FourCC &rttiFourCC) const

return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & *GetClassName* () const

get the class name

Util::FourCC *GetClassFourCC* () const

get the class FourCC code
<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>SetupGraphicsEntities</strong> ()</td>
<td>setup graphics entities</td>
</tr>
<tr>
<td>virtual void <strong>UpdateTransform</strong> (const Math::matrix44 &amp;m, bool setDirectly=false)</td>
<td>update the graphics entity's transform</td>
</tr>
<tr>
<td>void <strong>SetVisible</strong> (bool visible)</td>
<td>set the graphics entity's visibility</td>
</tr>
<tr>
<td>void <strong>OnSetOverwriteColor</strong> (const Ptr&lt; SetOverwriteColor &gt; &amp;msg)</td>
<td>on set overwrite color</td>
</tr>
<tr>
<td>void <strong>OnSetShaderVariable</strong> (const Ptr&lt; SetShaderVariable &gt; &amp;msg)</td>
<td>set shader variable</td>
</tr>
<tr>
<td>void <strong>SetEntity</strong> (const Ptr&lt; Entity &gt; &amp;v)</td>
<td>Set entity, this is attached to `v'.</td>
</tr>
<tr>
<td>void <strong>ClearEntity</strong> ()</td>
<td>Remove entity.</td>
</tr>
<tr>
<td>void <strong>RegisterMessage</strong> (const Id &amp;msgId)</td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

void GraphicsFeature::GraphicsProperty::OnActivate() [virtual, inherited]
called from Entity::ActivateProperties()
Attach the property to a game entity. This will create and setup the required graphics entities.
Reimplemented from Game::Property.
Reimplemented in StateObjectFeature::StateGraphicsProperty.

void GraphicsFeature::GraphicsProperty::OnDeactivate() [virtual, inherited]
called from Entity::DeactivateProperties()
Remove the property from its game entity. This will release the graphics entities owned by the property.
Reimplemented from Game::Property.
Reimplemented in StateObjectFeature::StateGraphicsProperty.

const Util::String & GraphicsFeature::GraphicsProperty::GetGraphicsResource() const [inline, virtual, inherited]
override to provide a self managed graphics resource (default is Attr::Graphics)
Get the default graphics resource, which is Attr::Graphics. subclasses may override this to provide a self managed resource.

void GraphicsFeature::GraphicsProperty::UpdateTransform(const Math::matrix44 & m, bool setDirectly = false) [protected,
update the graphics entity's transform

Called to update the graphics entity's transform.

```cpp
void GraphicsFeature::GraphicsProperty::SetVisible(bool visible) [protected, inherited]
```

Shows or hides all attached graphics entities.

```cpp
void Game::Property::OnLoad() [virtual, inherited]
```

called from within Entity::Load() after attributes are loaded

This method is called from within Game::Entity::Load() after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

```cpp
void Game::Property::OnStart() [virtual, inherited]
```

called from within Entity::OnStart() after OnLoad when the complete world exist

This method is called from within Game::Entity::OnStart(). This is the moment when the world is complete and the entity can establish connections to other entities.

Reimplemented in GraphicsFeature::CameraProperty.

```cpp
void Game::Property::OnSave() [virtual, inherited]
```
called from within Entity::Save() before attributes are saved back to database

This method is called from within Game::Entity::Save() before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

```cpp
void Game::Property::OnBeginFrame() [virtual, inherited]
```
called on begin of frame

This method is called from Game::Entity::OnBeginFrame() on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

```cpp
void Game::Property::OnMoveBefore() [virtual, inherited]
```
called before movement happens

This method is called from Game::Entity::OnMoveBefore() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before the physics subsystem is triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty.

```cpp
void Game::Property::OnMoveAfter() [virtual, inherited]
```
called after movement has happened
This method is called from `Game::Entity::OnMoveAfter()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`, and `PhysicsFeature::PhysicsProperty`.

```cpp
void Game::Property::OnRender() [virtual, inherited]
```
called before rendering happens

This method is called from `Game::Entity::OnRender()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before rendering happens.

Reimplemented in `GraphicsFeature::CameraProperty`, `GraphicsFeature::ChaseCameraProperty`, and `GraphicsFeature::MayaCameraProperty`.

```cpp
void Game::Property::OnLoseActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from `Game::Entity::OnLoseActivity()` on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`.

```cpp
void Game::Property::OnGainActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from `Game::Entity::OnRenderDebug()` on all
properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

```cpp
void Messaging::Port::AttachHandler (const Ptr<Handler> &h) [inherited]
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler (const Ptr<Handler> &h) [inherited]
```

remove a message handler from the port

Remove a message handler from the port.

```cpp
void Messaging::Port::Send (const Ptr<Message> &msg) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

const Util::FourCC & Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
GraphicsFeature::AnimationControlProperty
#include <animationcontrolproperty.h>

Inheritance diagram for GraphicsFeature::AnimationControlProperty:

- GraphicsFeature::AnimationControlProperty
- GraphicsFeature::GraphicsProperty
- Game::Property
- Messaging::Port
- Core::RefCounted
Detailed Description

Graphics property for animated characters.

(C) 2008 Radon Labs GmbH
Public Types

enum CallbackType
    callback types
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void SetupDefaultAttributes ()</td>
<td>setup default entity attributes</td>
</tr>
<tr>
<td>virtual void SetupAcceptedMessages ()</td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>virtual void HandleMessage (const Ptr&lt;Message&gt;&amp; msg)</td>
<td>handle a single message</td>
</tr>
<tr>
<td>virtual void OnActivate ()</td>
<td>called from Entity::ActivateProperties()</td>
</tr>
<tr>
<td>virtual void OnDeactivate ()</td>
<td>called from Entity::DeactivateProperties()</td>
</tr>
<tr>
<td>virtual void SetupCallbacks ()</td>
<td>setup callbacks for this property, call by entity in OnActivate()</td>
</tr>
<tr>
<td>virtual const Util::String &amp; GetGraphicsResource () const</td>
<td>override to provide a self managed graphics resource (default is Attr::Graphics)</td>
</tr>
<tr>
<td>const Ptr&lt;Entity&gt; &amp; GetEntity () const</td>
<td>get entity this property is attached to</td>
</tr>
<tr>
<td>bool HasEntity () const</td>
<td>return true if entity pointer is valid</td>
</tr>
<tr>
<td>bool IsActive () const</td>
<td>return true if property is currently active</td>
</tr>
<tr>
<td>virtual void OnLoad ()</td>
<td>called from within Entity::Load() after attributes are loaded</td>
</tr>
<tr>
<td>virtual void OnStart ()</td>
<td>called from within Entity::OnStart() after OnLoad when the complete world exist</td>
</tr>
<tr>
<td>virtual void OnSave ()</td>
<td>called from within Entity::Save() before attributes are saved back to database</td>
</tr>
<tr>
<td>virtual void OnBeginFrame ()</td>
<td></td>
</tr>
</tbody>
</table>
called on begin of frame

virtual void **OnMoveBefore** ()
called before movement happens

virtual void **OnMoveAfter** ()
called after movement has happened

virtual void **OnRender** ()
called before rendering happens

virtual void **OnLoseActivity** ()
called when game debug visualization is on

virtual void **OnGainActivity** ()
called when game debug visualization is on

void **AttachHandler** (const **Ptr**< **Handler** > &h)
attach a message handler to the port

void **RemoveHandler** (const **Ptr**< **Handler** > &h)
remove a message handler from the port

void **RemoveAllHandlers** ()
remove all message handler from the port

**SizeT** **GetNumHandlers** () const
return number of handlers attached to the port

const **Ptr**< **Handler** > & **GetHandlerAtIndex** (IndexT i) const
get a message handler by index

virtual void **Send** (const **Ptr**< **Message** > &msg)
send a message to the port

const **Util::Array**< const **Id** * > & **GetAcceptedMessages** () const
get the array of accepted messages (sorted)

bool **AcceptsMessage** (const **Id** &msgId) const
return true if port accepts this msg

int **GetRefCount** () const
get the current refcount

void **AddRef** ()
increment refcount by one

void **Release** ()
decrement refcount and destroy object if refcount is zero

bool **IsInstanceOf** (const **Rtti** &rtti) const
return true if this object is instance of given class
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void DumpRefCountingLeaks ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void</td>
<td><strong>SetupGraphicsEntities</strong> ()</td>
<td>setup graphics entities</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>UpdateTransform</strong> (const Math::matrix44 &amp;m, bool setDirectly=false)</td>
<td>update the graphics entity's transform</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetVisible</strong> (bool visible)</td>
<td>set visible</td>
</tr>
<tr>
<td>void</td>
<td><strong>OnSetOverwriteColor</strong> (const Ptr&lt; SetOverwriteColor &gt; &amp;msg)</td>
<td>on set overwrite color</td>
</tr>
<tr>
<td>void</td>
<td><strong>OnSetShaderVariable</strong> (const Ptr&lt; SetShaderVariable &gt; &amp;msg)</td>
<td>set shader variable</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetEntity</strong> (const Ptr&lt; Entity &gt; &amp;v)</td>
<td>Set entity, this is attached to <code>v</code>.</td>
</tr>
<tr>
<td>void</td>
<td><strong>ClearEntity</strong> ()</td>
<td>Remove entity.</td>
</tr>
<tr>
<td>void</td>
<td><strong>RegisterMessage</strong> (const Id &amp;msgId)</td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void GraphicsFeature::GraphicsProperty::OnActivate ( ) [virtual, inherited]

called from Entity::ActivateProperties()

Attach the property to a game entity. This will create and setup the required graphics entities.

Reimplemented from Game::Property.

Reimplemented in StateObjectFeature::StateGraphicsProperty.

```cpp
void GraphicsFeature::GraphicsProperty::OnDeactivate ( ) [virtual, inherited]

called from Entity::DeactivateProperties()

Remove the property from its game entity. This will release the graphics entities owned by the property.

Reimplemented from Game::Property.

Reimplemented in StateObjectFeature::StateGraphicsProperty.

```cpp
const Util::String & GraphicsFeature::GraphicsProperty::GetGraphicsResource ( ) const [inline, virtual, inherited]

override to provide a self managed graphics resource (default is Attr::Graphics)

Get the default graphics resource, which is Attr::Graphics. subclasses may override this to provide a self managed resource.

```cpp
void GraphicsFeature::GraphicsProperty::SetupGraphicsEntities ( ) [protected, virtual, inherited]

setup graphics entities
```
Setup the graphics entities. You may override this method in a subclass if different setup is needed.

Reimplemented in GraphicsFeature::ActorGraphicsProperty.

```cpp
void GraphicsFeature::GraphicsProperty::UpdateTransform(
    const Math::matrix44 &m,
    bool setDirectly = false)
[protected, virtual, inherited]
```

update the graphics entity's transform

Called to update the graphics entity's transform.

```cpp
void GraphicsFeature::GraphicsProperty::SetVisible(bool visible)
[protected, inherited]
```

Shows or hides all attached graphics entities.

```cpp
void Game::Property::OnLoad()
[virtual, inherited]
```

called from within Entity::Load() after attributes are loaded

This method is called from within Game::Entity::Load() after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

```cpp
void Game::Property::OnStart()
[virtual, inherited]
```

called from within Entity::OnStart() after OnLoad when the complete world exist
This method is called from within `Game::Entity::OnStart()`. This is the moment when the world is complete and the entity can establish connections to other entities.

Reimplemented in `GraphicsFeature::CameraProperty`.

```c++
void Game::Property::OnSave() [virtual, inherited]
```
called from within `Entity::Save()` before attributes are saved back to database

This method is called from within `Game::Entity::Save()` before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in `PhysicsFeature::TriggerProperty`, and `StateObjectFeature::StateProperty`.

```c++
void Game::Property::OnBeginFrame() [virtual, inherited]
```
called on begin of frame

This method is called from `Game::Entity::OnBeginFrame()` on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in `PhysicsFeature::TriggerProperty`, and `StateObjectFeature::StateProperty`.

```c++
void Game::Property::OnMoveBefore() [virtual, inherited]
```
called before movement happens

This method is called from `Game::Entity::OnMoveBefore()` on all properties attached to an entity in the order of attachment. Override
this method if your property has any work to do before the physics subsystem is triggered.

Reimplemented in **PhysicsFeature::ActorPhysicsProperty**.

```cpp
void Game::Property::OnMoveAfter() [virtual, inherited]
```
called after movement has happened

This method is called from **Game::Entity::OnMoveAfter()** on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in **PhysicsFeature::ActorPhysicsProperty**, and **PhysicsFeature::PhysicsProperty**.

```cpp
void Game::Property::OnRender() [virtual, inherited]
```
called before rendering happens

This method is called from **Game::Entity::OnRender()** on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before rendering happens.

Reimplemented in **GraphicsFeature::CameraProperty**, **GraphicsFeature::ChaseCameraProperty**, and **GraphicsFeature::MayaCameraProperty**.

```cpp
void Game::Property::OnLoseActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from **Game::Entity::OnLoseActivity()** on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active
Reimplemented in \texttt{PhysicsFeature::ActorPhysicsProperty}.

\begin{verbatim}
void Game::Property::OnGainActivity( ) [virtual, inherited]
\end{verbatim}
called when game debug visualization is on

This method is called from \texttt{Game::Entity::OnRenderDebug()} on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

\begin{verbatim}
void Messaging::Port::AttachHandler( const Ptr<Handler> & h ) [inherited]
\end{verbatim}
attach a message handler to the port

\begin{verbatim}
void Messaging::Port::RemoveHandler( const Ptr<Handler> & h ) [inherited]
\end{verbatim}
remove a message handler from the port

\begin{verbatim}
void Messaging::Port::Send( const Ptr<Message> & msg ) [virtual, inherited]
\end{verbatim}
send a message to the port

Send a message to the port. This will immediately call the \texttt{HandleMessage()} method of all attached handlers. If the message has been handled by at least one of the handlers, the \texttt{Handled()} flag
of the message will be set to true.

int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

going current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
GraphicsFeature::AttachmentManager
GraphicsFeature::AttachmentManager
Class Reference

#include <attachmentmanager.h>

Inheritance diagram for GraphicsFeature::AttachmentManager:
Detailed Description

Main thread side manager (frontend) for managing attachments.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AttachmentManager ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~AttachmentManager ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>OnFrame ()</strong></td>
<td>Trigger</td>
</tr>
<tr>
<td><strong>AddGraphicsAttachment</strong></td>
<td>Add a attachment</td>
</tr>
</tbody>
</table>
| (const Util::StringAtom &jointName,
  const Util::StringAtom &baseEntityId,
  const Util::StringAtom &entityToAttachResId,
  const Math::matrix44 &offset,
  bool keepLocal,
  InternalGraphics::AttachmentServer::AttachmentRotation rotation) |                                                                            |
| **AddGraphicsAttachmentTemporary**            | Attach a attachment temporary                                               |
| (const Util::StringAtom &jointName,
  const Util::StringAtom &baseEntityId,
  const Util::StringAtom &entityToAttachResId,
  const Math::matrix44 &offset,
  bool keepLocal,
  InternalGraphics::AttachmentServer::AttachmentRotation rotation,
  Timing::Time duration) |                                                                            |
| **AddGraphicsAttachment**                     | Add a attachment                                                           |
| (const Util::StringAtom &jointName,
  const Ptr<Game::Entity> &baseEntityPtr,
  const Util::StringAtom &entityToAttachResId,
  const Math::matrix44 &offset,
  bool keepLocal,
  InternalGraphics::AttachmentServer::AttachmentRotation rotation) |                                                                            |
| **AddGraphicsAttachmentTemporary**            | Attach a attachment temporary                                               |
| (const Util::StringAtom &jointName,
  const Ptr<Game::Entity> &baseEntityPtr,
  const Util::StringAtom &entityToAttachResId,
  const Math::matrix44 &offset,
  bool keepLocal,
  InternalGraphics::AttachmentServer::AttachmentRotation rotation,
  Timing::Time duration) |                                                                            |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void ClearAttachments ()</code></td>
<td>Clear all attachments</td>
</tr>
<tr>
<td><code>void ClearAttachmentsOnEntity (const Ptr&lt; Game &amp;baseEntity)</code></td>
<td>Remove all attachments on given base entity</td>
</tr>
<tr>
<td><code>virtual void OnActivate ()</code></td>
<td>Called when attached to game server</td>
</tr>
<tr>
<td><code>virtual void OnDeactivate ()</code></td>
<td>Called when removed from game server</td>
</tr>
<tr>
<td><code>bool IsActive () const</code></td>
<td>Return true if currently active</td>
</tr>
<tr>
<td><code>virtual void OnBeginFrame ()</code></td>
<td>Called before frame by the game server</td>
</tr>
<tr>
<td><code>virtual void OnEndFrame ()</code></td>
<td>Called after frame by the game server</td>
</tr>
<tr>
<td><code>virtual void OnLoad ()</code></td>
<td>Called after loading game state</td>
</tr>
<tr>
<td><code>virtual void OnSave ()</code></td>
<td>Called before saving game state</td>
</tr>
<tr>
<td><code>virtual void OnStart ()</code></td>
<td>Called by Game::Server::Start() when the world is started</td>
</tr>
<tr>
<td><code>virtual void OnRenderDebug ()</code></td>
<td>Render a debug visualization</td>
</tr>
<tr>
<td><code>virtual void HandleMessage (const Ptr&lt; Messaging::Message &amp;message) </code></td>
<td>Handle a single message (distribute to ports which accept the message)</td>
</tr>
<tr>
<td><code>void AttachPort (const Ptr&lt; Port &gt; &amp;port)</code></td>
<td>Attach a message port</td>
</tr>
<tr>
<td><code>void RemovePort (const Ptr&lt; Port &gt; &amp;port)</code></td>
<td>Remove a message port</td>
</tr>
<tr>
<td><code>bool HasPort (const Ptr&lt; Port &gt; &amp;port) const</code></td>
<td>Return true if a port exists</td>
</tr>
<tr>
<td><code>virtual void SetupAcceptedMessages ()</code></td>
<td>Override to register accepted messages</td>
</tr>
<tr>
<td><code>void AttachHandler (const Ptr&lt; Handler &gt; &amp;h)</code></td>
<td>Attach a message handler to the port</td>
</tr>
</tbody>
</table>

*Note: The comments have been added for clarity.*
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RemoveHandler (const Ptr&lt; Handler &gt; &amp;h)</code></td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td><code>void RemoveAllHandlers ()</code></td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td><code>SizeT GetNumHandlers () const</code></td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td><code>const Ptr&lt; Handler &gt; &amp; GetHandlerAtIndex (IndexT i) const</code></td>
<td>get a message handler by index</td>
</tr>
<tr>
<td><code>virtual void Send (const Ptr&lt; Message &gt; &amp;msg)</code></td>
<td>send a message to the port</td>
</tr>
<tr>
<td><code>const Util::Array&lt; const Id * &gt; &amp; GetAcceptedMessages () const</code></td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td><code>bool AcceptsMessage (const Id &amp;msgId) const</code></td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
void RegisterMessage (const Id &msgId)

register a single accepted message
```
Member Function Documentation

void Game::Manager::OnActivate() [virtual, inherited]
called when attached to game server

This method is called when the manager is attached to the game server. The manager base class will register its message port with the message server.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager, BaseGameFeature::EnvQueryManager, BaseGameFeature::GlobalAttrsManager, Script::DialogManager, and Script::ScriptManager.

void Game::Manager::OnDeactivate() [virtual, inherited]
called when removed from game server

This method is called when the manager is removed from the game server. It will unregister its message port from the message server at this point.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager, BaseGameFeature::EntityManager, BaseGameFeature::EnvEntityManager, BaseGameFeature::EnvQueryManager, Script::DialogManager, and Script::ScriptManager.

void Game::Manager::OnBeginFrame() [virtual, inherited]
called before frame by the game server

Called before frame, override in subclasses
Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Game::Manager::OnEndFrame() [virtual, inherited]
```
called after frame by the game server

Called after frame, override in subclasses

Reimplemented in **BaseGameFeature::EntityManager**.

```cpp
void Messaging::Dispatcher::HandleMessage(const Messaging::Message msg) [virtual, inherited]
```
handle a single message (distribute to ports which accept the message)

Handle a message. The message will only be distributed to ports which accept the message.

Reimplemented from **Messaging::Port**.

Reimplemented in **Script::DialogManager**.

```cpp
void Messaging::Dispatcher::AttachPort(const Port & port) [inherited]
```
attach a message port

Attach a new message port.

**Parameters:**

- `port` pointer to a message port object

```cpp
void Messaging::Dispatcher::RemovePort(const Port & port) [inherited]
```
remove a message port
Remove a message port object.

**Parameters:**

- *handler* pointer to message port object to be removed

```cpp
bool Messaging::Dispatcher::HasPort(const Ptr<Port>& port) const [inherited]
```

return true if a port exists

Return true if a port is already attached.

```cpp
void Messaging::Port::AttachHandler(const Ptr<Handler>& h) [inherited]
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler(const Ptr<Handler>& h) [inherited]
```

remove a message handler from the port

Remove a message handler from the port.

```cpp
void Messaging::Port::Send(const Ptr<Message>& msg) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.
int Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:45 2010
GraphicsFeature::CameraProperty
GraphicsFeature::CameraProperty
Class Reference

#include <cameraproperty.h>

Inheritance diagram for GraphicsFeature::CameraProperty:
Detailed Description

A camera property adds the ability to manipulate the camera to an entity. Please note that more advanced camera properties should always be derived from the class camera property if camera focus handling is desired, since the FocusManager will only work on game entities which have a `CameraProperty` (or a subclass) attached.

It is completely ok though to handle camera manipulation in a property not derived from `CameraProperty`, but please be aware that the FocusManager will ignore those.

The camera property will generally

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>CallbackType</th>
</tr>
</thead>
</table>

*callback types*
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CameraProperty()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~CameraProperty()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>OnDeactivate()</code></td>
<td>Called from <code>Entity::DeactivateProperties()</code></td>
</tr>
<tr>
<td><code>SetupAcceptedMessages()</code></td>
<td>Setup accepted messages</td>
</tr>
<tr>
<td><code>SetupCallbacks()</code></td>
<td>Setup callbacks for this property</td>
</tr>
<tr>
<td><code>SetupDefaultAttributes()</code></td>
<td>Setup default entity attributes</td>
</tr>
<tr>
<td><code>OnStart()</code></td>
<td>Called from within <code>Entity::OnStart()</code> after <code>OnLoad</code> when the complete world exist</td>
</tr>
<tr>
<td><code>OnObtainCameraFocus()</code></td>
<td>Called when camera focus is obtained</td>
</tr>
<tr>
<td><code>OnLoseCameraFocus()</code></td>
<td>Called when camera focus is lost</td>
</tr>
<tr>
<td><code>OnRender()</code></td>
<td>Called before rendering happens</td>
</tr>
<tr>
<td><code>HasFocus()</code></td>
<td>Return true if currently has camera focus</td>
</tr>
<tr>
<td><code>HandleMessage(const Ptr&lt;Messaging::Message&gt; &amp; msg)</code></td>
<td>Handle a single message</td>
</tr>
<tr>
<td><code>GetEntity()</code></td>
<td>Get entity this property is attached to</td>
</tr>
<tr>
<td><code>HasEntity()</code></td>
<td>Return true if entity pointer is valid</td>
</tr>
<tr>
<td><code>OnActivate()</code></td>
<td>Called from <code>Entity::ActivateProperties()</code></td>
</tr>
<tr>
<td><code>IsActive()</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>OnLoad</strong>()</td>
<td>Called from within Entity::Load() after attributes are loaded</td>
</tr>
<tr>
<td><strong>OnSave</strong>()</td>
<td>Called from within Entity::Save() before attributes are saved back to database</td>
</tr>
<tr>
<td><strong>OnBeginFrame</strong>()</td>
<td>Called on begin of frame</td>
</tr>
<tr>
<td><strong>OnMoveBefore</strong>()</td>
<td>Called before movement happens</td>
</tr>
<tr>
<td><strong>OnMoveAfter</strong>()</td>
<td>Called after movement has happened</td>
</tr>
<tr>
<td><strong>OnRenderDebug</strong>()</td>
<td>Called when game debug visualization is on</td>
</tr>
<tr>
<td><strong>OnLoseActivity</strong>()</td>
<td>Called when game debug visualization is on</td>
</tr>
<tr>
<td><strong>OnGainActivity</strong>()</td>
<td>Called when game debug visualization is on</td>
</tr>
<tr>
<td><strong>AttachHandler</strong> (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>Attach a message handler to the port</td>
</tr>
<tr>
<td><strong>RemoveHandler</strong> (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>Remove a message handler from the port</td>
</tr>
<tr>
<td><strong>RemoveAllHandlers</strong>()</td>
<td>Remove all message handler from the port</td>
</tr>
<tr>
<td><strong>GetNumHandlers</strong>() const</td>
<td>Return number of handlers attached to the port</td>
</tr>
<tr>
<td><strong>GetHandlerAtIndex</strong> (IndexT i) const</td>
<td>Get a message handler by index</td>
</tr>
<tr>
<td><strong>Send</strong> (const Ptr&lt; Message &gt; &amp;msg)</td>
<td>Send a message to the port</td>
</tr>
<tr>
<td><strong>GetAcceptedMessages</strong>() const</td>
<td>Get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td><strong>AcceptsMessage</strong> (const Id &amp;msgId) const</td>
<td>Return true if port accepts this msg</td>
</tr>
<tr>
<td><strong>GetRefCount</strong>() const</td>
<td>Get the reference count of the port</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>bool IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>bool IsA (const Util::String &amp;rttiName) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>bool IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetClassName () const</strong></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC GetClassFourCC () const</strong></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application</em> <em>(NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
# Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void UpdateAudioListenerPosition()</code> const</td>
<td>update audio listener position</td>
</tr>
<tr>
<td><code>void SetEntity(const Ptr&lt; Entity &gt; &amp;v)</code></td>
<td>Set entity, this is attached to, to <code>v</code>.</td>
</tr>
<tr>
<td><code>void ClearEntity()</code></td>
<td>Remove entity.</td>
</tr>
<tr>
<td><code>void RegisterMessage(const Id &amp;msgId)</code></td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
void GraphicsFeature::CameraProperty::OnObtainCameraFocus() [virtual]
called when camera focus is obtained

This method is called by the FocusManager when our entity gains the camera focus. Override this method if your subclass needs to do some initialization when gaining the camera focus.

Reimplemented in GraphicsFeature::ChaseCameraProperty.

void GraphicsFeature::CameraProperty::OnLoseCameraFocus() [virtual]
called when camera focus is lost

This method is called by the FocusManager when our entity loses the camera focus. Override this method if your subclass needs to do any cleanup work here.

void GraphicsFeature::CameraProperty::OnRender() [virtual]
called before rendering happens

Called before camera is "rendered". The default camera properties applies shake effects to the camera.

Reimplemented from Game::Property.

Reimplemented in GraphicsFeature::ChaseCameraProperty, and GraphicsFeature::MayaCameraProperty.

bool GraphicsFeature::CameraProperty::HasFocus() const [virtual]

return true if currently has camera focus
This method returns true if our entity has the camera focus. This implementation makes sure that 2 properties cannot report that they have the camera focus by accident.

```cpp
void Game::Property::OnActivate() [virtual, inherited]
```
called from Entity::ActivateProperties()

This method is called by Game::Entity::ActivateProperties(). Use this method for one-time initializations of the property.


```cpp
void Game::Property::OnLoad() [virtual, inherited]
```
called from within Entity::Load() after attributes are loaded

This method is called from within Game::Entity::Load() after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

```cpp
void Game::Property::OnSave() [virtual, inherited]
```
called from within Entity::Save() before attributes are saved back to database
This method is called from within Game::Entity::Save() before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

```cpp
void Game::Property::OnBeginFrame() [virtual, inherited]
```
called on begin of frame

This method is called from Game::Entity::OnBeginFrame() on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

```cpp
void Game::Property::OnMoveBefore() [virtual, inherited]
```
called before movement happens

This method is called from Game::Entity::OnMoveBefore() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before the physics subsystem is triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty.

```cpp
void Game::Property::OnMoveAfter() [virtual, inherited]
```
called after movement has happened

This method is called from Game::Entity::OnMoveAfter() on all properties attached to an entity in the order of attachment. Override
this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty, and PhysicsFeature::PhysicsProperty.

```cpp
void Game::Property::OnRenderDebug() [virtual, inherited]
```
called when game debug visualization is on

This method is called from Game::Entity::OnRenderDebug() on all properties attached to an entity in the order of attachment. It's meant for debug issues. It will be called when debug mode is enabled.

Reimplemented in GraphicsFeature::GraphicsProperty, PhysicsFeature::ActorPhysicsProperty, and PhysicsFeature::TriggerProperty.

```cpp
void Game::Property::OnLoseActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from Game::Entity::OnLoseActivity() on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

Reimplemented in PhysicsFeature::ActorPhysicsProperty.

```cpp
void Game::Property::OnGainActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from Game::Entity::OnRenderDebug() on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active
camera)

```cpp
void Messaging::Port::AttachHandler(const Ptr<Handler>& h) [inherited]
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler(const Ptr<Handler>& h) [inherited]
```

remove a message handler from the port

Remove a message handler from the port.

```cpp
void Messaging::Port::Send(const Ptr<Message>& msg) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the Handled() flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

GraphicsFeature::ChaseCameraProperty
GraphicsFeature::ChaseCameraProperty
Class Reference

#include <chasecameraproperty.h>

Inheritance diagram for GraphicsFeature::ChaseCameraProperty:
Detailed Description

A chase camera for 3rd person camera control.

(C) 2005 Radon Labs GmbH
Public Types

enum CallbackType

callback types
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ChaseCameraProperty</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~ChaseCameraProperty</strong> ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>SetupDefaultAttributes</strong> ()</td>
<td>Setup default entity attributes</td>
</tr>
<tr>
<td>virtual void <strong>OnActivate</strong> ()</td>
<td>Called from <code>Entity::ActivateProperties()</code></td>
</tr>
<tr>
<td>virtual void <strong>OnObtainCameraFocus</strong> ()</td>
<td>Called when camera focus is obtained</td>
</tr>
<tr>
<td>virtual void <strong>OnRender</strong> ()</td>
<td>Called before camera is &quot;rendered&quot;</td>
</tr>
<tr>
<td>virtual void <strong>SetupAcceptedMessages</strong> ()</td>
<td>Override to register accepted messages</td>
</tr>
<tr>
<td>virtual void <strong>HandleMessage</strong> (const <strong>Ptr</strong>&lt; <strong>Messaging::Message</strong> &gt; &amp;msg)</td>
<td>Handle a single message</td>
</tr>
<tr>
<td>virtual void <strong>OnDeactivate</strong> ()</td>
<td>Called from <code>Entity::DeactivateProperties()</code></td>
</tr>
<tr>
<td>virtual void <strong>SetupCallbacks</strong> ()</td>
<td>Setup callbacks for this property</td>
</tr>
<tr>
<td>virtual void <strong>OnStart</strong> ()</td>
<td>Called from within <code>Entity::OnStart()</code> after <code>OnLoad</code> when the complete world exist</td>
</tr>
<tr>
<td>virtual void <strong>OnLoseCameraFocus</strong> ()</td>
<td>Called when camera focus is lost</td>
</tr>
<tr>
<td>virtual bool <strong>HasFocus</strong> () const</td>
<td>Return true if currently has camera focus</td>
</tr>
<tr>
<td>const <strong>Ptr</strong>&lt; <strong>Entity</strong> &gt; &amp; <strong>GetEntity</strong> () const</td>
<td>Get entity this property is attached to</td>
</tr>
<tr>
<td>bool <strong>HasEntity</strong> () const</td>
<td>Return true if entity pointer is valid</td>
</tr>
<tr>
<td>bool <strong>IsActive</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>**virtual void ** OnLoad ()</td>
<td>called from within Entity::Load() after attributes are loaded</td>
</tr>
<tr>
<td>**virtual void ** OnSave ()</td>
<td>called from within Entity::Save() before attributes are saved back to database</td>
</tr>
<tr>
<td>**virtual void ** OnBeginFrame ()</td>
<td>called on begin of frame</td>
</tr>
<tr>
<td>**virtual void ** OnMoveBefore ()</td>
<td>called before movement happens</td>
</tr>
<tr>
<td>**virtual void ** OnMoveAfter ()</td>
<td>called after movement has happened</td>
</tr>
<tr>
<td>**virtual void ** OnRenderDebug ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>**virtual void ** OnLoseActivity ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>**virtual void ** OnGainActivity ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>void AttachHandler (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td>void RemoveHandler (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td>void RemoveAllHandlers ()</td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td>SizeT GetNumHandlers () const</td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td>const Ptr&lt; Handler &gt; &amp; GetHandlerAtIndex (IndexT i) const</td>
<td>get a message handler by index</td>
</tr>
<tr>
<td>**virtual void ** Send (const Ptr&lt; Message &gt; &amp;msg)</td>
<td>send a message to the port</td>
</tr>
<tr>
<td>const Util::Array&lt; const Id * &gt; &amp; GetAcceptedMessages () const</td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td>bool AcceptsMessage (const Id &amp;msgId) const</td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>get the current refcount</td>
<td></td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual <code>Math::point</code> <strong>DoCollideCheck</strong> (const <code>Math::point</code> &amp;from, const <code>Math::point</code> &amp;to)</td>
<td>do a collision check</td>
</tr>
<tr>
<td>virtual void HandleCameraReset ()</td>
<td>handle a camera reset message</td>
</tr>
<tr>
<td>virtual void HandleCameraOrbit (float dRho, float dTheta)</td>
<td>handle a camera orbit message</td>
</tr>
<tr>
<td>virtual void HandleCameraDistanceChange (float z)</td>
<td>handle a camera distance change</td>
</tr>
<tr>
<td>virtual void UpdateCamera (bool interpolate)</td>
<td>update the camera matrix</td>
</tr>
<tr>
<td>void UpdateAudioListenerPosition () const</td>
<td>update audio listener position</td>
</tr>
<tr>
<td>void SetEntity (const Ptr&lt; Entity &gt; &amp;v)</td>
<td>Set entity, this is attached to, to <code>v</code>.</td>
</tr>
<tr>
<td>void ClearEntity ()</td>
<td>Remove entity.</td>
</tr>
<tr>
<td>void RegisterMessage (const Id &amp;msgId)</td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

void GraphicsFeature::ChaseCameraProperty::SetupDefaultAttributes() [virtual]
setup default entity attributes
This adds the default attributes to the property.
Reimplemented from GraphicsFeature::CameraProperty.

void GraphicsFeature::ChaseCameraProperty::OnActivate() [virtual]
called from Entity::ActivateProperties()
This method is called once when the property is attached to the entity.
Reimplemented from Game::Property.

void GraphicsFeature::ChaseCameraProperty::OnObtainCameraFocus() [virtual]
called when camera focus is obtained
This method is called when the entity obtains the camera focus.
Reimplemented from GraphicsFeature::CameraProperty.

void GraphicsFeature::ChaseCameraProperty::OnRender() [virtual]
called before camera is "rendered"
This method is called before the camera is "rendered".
Reimplemented from GraphicsFeature::CameraProperty.

void GraphicsFeature::ChaseCameraProperty::HandleMessage(const ::Messaging::Message msg) [virtual]
handle a single message

This method handles pending messages for the property. It is called by the `OnRender()` method.

Reimplemented from `GraphicsFeature::CameraProperty`.

```cpp
point
GraphicsFeature::ChaseCameraProperty::DoCollideCheck ( const Math::point from, 
&
const Math::point to 
&
) [protected, virtual]
```

do a collision check

Do a ray check between 'from' and 'to' and return a replacement point for 'to'.

```cpp
void
GraphicsFeature::ChaseCameraProperty::HandleCameraReset () [protected, virtual]
```

handle a camera reset message

Handle a camera reset.

```cpp
void
GraphicsFeature::ChaseCameraProperty::HandleCameraOrbit ( float dRho, 
float dTheta 
) [protected, virtual]
```

handle a camera orbit message

Handle a camera orbit.

```cpp
void
GraphicsFeature::ChaseCameraProperty::HandleCameraDistanceChange ( float d ) [protected, virtual]
```

handle a camera distance change

Handle a camera distance change.
void GraphicsFeature::ChaseCameraProperty::UpdateCamera ( bool interpolate ) [protected, virtual]

update the camera matrix

Update the camera position and orientation from the current orbit polar angles.

void GraphicsFeature::CameraProperty::OnLoseCameraFocus ( ) [virtual, inherited]
called when camera focus is lost

This method is called by the FocusManager when our entity loses the camera focus. Override this method if your subclass needs to do any cleanup work here.

bool GraphicsFeature::CameraProperty::HasFocus ( ) const [virtual, inherited]

return true if currently has camera focus

This method returns true if our entity has the camera focus. This implementation makes sure that 2 properties cannot report that they have the camera focus by accident.

void Game::Property::OnLoad ( ) [virtual, inherited]
called from within Entity::Load() after attributes are loaded

This method is called from within Game::Entity::Load() after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void
Game::Property::OnSave ( ) [virtual, inherited]

called from within Entity::Save() before attributes are saved back to database

This method is called from within Game::Entity::Save() before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void
Game::Property::OnBeginFrame ( ) [virtual, inherited]

called on begin of frame

This method is called from Game::Entity::OnBeginFrame() on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void
Game::Property::OnMoveBefore ( ) [virtual, inherited]

called before movement happens

This method is called from Game::Entity::OnMoveBefore() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before the physics subsystem is triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty.

void
Game::Property::OnMoveAfter ( ) [virtual, inherited]
called after movement has happened

This method is called from Game::Entity::OnMoveAfter() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty, and PhysicsFeature::PhysicsProperty.

void Game::Property::OnRenderDebug() [virtual, inherited]
called when game debug visualization is on

This method is called from Game::Entity::OnRenderDebug() on all properties attached to an entity in the order of attachment. It's meant for debug issues. It will be called when debug mode is enabled.

Reimplemented in GraphicsFeature::GraphicsProperty, PhysicsFeature::ActorPhysicsProperty, and PhysicsFeature::TriggerProperty.

void Game::Property::OnLoseActivity() [virtual, inherited]
called when game debug visualization is on

This method is called from Game::Entity::OnLoseActivity() on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

Reimplemented in PhysicsFeature::ActorPhysicsProperty.

void Game::Property::OnGainActivity() [virtual, inherited]
called when game debug visualization is on
This method is called from `Game::Entity::OnRenderDebug()` on all properties attached to an entity in the order of attachment. It indicates that the entity will be triggered from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

```cpp
void Messaging::Port::AttachHandler (const Ptr<Handler> & h ) [inherited]
attach a message handler to the port

void Messaging::Port::RemoveHandler (const Ptr<Handler> & h ) [inherited]
remove a message handler from the port

void Messaging::Port::Send (const Ptr<Message> & msg ) [virtual, inherited]
send a message to the port
```

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
get the current refcount
```

Return the current refcount of the object.
increment refcount by one

Increment the refcount of the object.

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

get the class name

Get the class name of the object.

get the class FourCC code

Get the class FourCC of the object.

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
GraphicsFeature::GraphicsFeatureUnitUnit
GraphicsFeature::GraphicsFeatureUnit

Class Reference

#include <graphicsfeatureunit.h>
Detailed Description

The GraphicsFeatures provides everything for rendering graphic entities from different views in different stages.

For the default use it creates one default view and one default stage which are used in the graphic and camera properties. The BaseGameFeatureUnit uses this features to build up a default graphic world.

Additional to the rendering the graphics feature creates and triggers the input server.

(C) 2007 Radon Labs GmbH
GraphicsFeature::GraphicsProperty
GraphicsFeature::GraphicsProperty
Class Reference

#include <graphicsproperty.h>

Inheritance diagram for GraphicsFeature::GraphicsProperty:
Detailed Description

This is the standard graphics property which adds visibility to a game entity.

NOTE: There are cases where the graphics property may depend on a physics property (for complex physics entities which require several graphics entities to render themselves). Thus it is recommended that physics properties are attached before graphics properties.

(C) 2007 Radon Labs GmbH
# Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>CallbackType</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback types</td>
<td></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Member Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GraphicsProperty()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~GraphicsProperty()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <code>SetupDefaultAttributes()</code></td>
<td>setup default entity attributes</td>
</tr>
<tr>
<td>virtual void <code>OnActivate()</code></td>
<td>called from <code>Entity::ActivateProperties()</code></td>
</tr>
<tr>
<td>virtual void <code>OnDeactivate()</code></td>
<td>called from <code>Entity::DeactivateProperties()</code></td>
</tr>
<tr>
<td>virtual void <code>SetupCallbacks()</code></td>
<td>setup callbacks for this property, call by entity in <code>OnActivate()</code></td>
</tr>
<tr>
<td>virtual void <code>SetupAcceptedMessages()</code></td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>virtual void <code>HandleMessage(const </code>Ptr&lt; <code>Messaging::Message</code>&gt; &amp;msg)`</td>
<td>handle a single message</td>
</tr>
<tr>
<td>virtual const <code>Util::String &amp;</code> <code>GetGraphicsResource()</code></td>
<td>override to provide a self managed graphics resource (default is <code>Attr::Graphics</code>)</td>
</tr>
<tr>
<td>virtual void <code>OnRenderDebug()</code></td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>const <code>Ptr&lt; </code>Entity<code>&gt; &amp;</code> <code>GetEntity()</code></td>
<td>get entity this property is attached to</td>
</tr>
<tr>
<td>bool <code>HasEntity()</code></td>
<td>return true if entity pointer is valid</td>
</tr>
<tr>
<td>bool <code>IsActive()</code></td>
<td>return true if property is currently active</td>
</tr>
<tr>
<td>virtual void <code>OnLoad()</code></td>
<td>called from within <code>Entity::Load()</code> after attributes are loaded</td>
</tr>
<tr>
<td>virtual void <code>OnStart()</code></td>
<td>called from within <code>Entity::OnStart()</code> after <code>OnLoad</code> when</td>
</tr>
</tbody>
</table>
the complete world exist

virtual void **OnSave** ()
called from within Entity::Save() before attributes are saved back to database

virtual void **OnBeginFrame** ()
called on begin of frame

virtual void **OnMoveBefore** ()
called before movement happens

virtual void **OnMoveAfter** ()
called after movement has happened

virtual void **OnRender** ()
called before rendering happens

virtual void **OnLoseActivity** ()
called when game debug visualization is on

virtual void **OnGainActivity** ()
called when game debug visualization is on

**void** AttachHandler (const **Ptr<Handler>** &h)
attach a message handler to the port

**void** RemoveHandler (const **Ptr<Handler>** &h)
remove a message handler from the port

**void** RemoveAllHandlers ()
remove all message handler from the port

**SizeT** GetNumHandlers () const
return number of handlers attached to the port

const **Ptr<Handler>** & **GetHandlerAtIndex** (IndexT i) const
get a message handler by index

virtual void **Send** (const **Ptr<Message>** &msg)
send a message to the port

const **Util::Array< const Id * >** & **GetAcceptedMessages** () const
get the array of accepted messages (sorted)

**bool** AcceptsMessage (const **Id** &msgId) const
return true if port accepts this msg

**int** GetRefCount () const
get the current refcount

**void** AddRef ()
increment refcount by one
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName() const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC() const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void SetupGraphicsEntities()</td>
<td>setup graphics entities</td>
</tr>
<tr>
<td>virtual void UpdateTransform (const Math::matrix44 &amp;m, bool setDirectly=false)</td>
<td>update the graphics entity's transform</td>
</tr>
<tr>
<td>void SetVisible (bool visible)</td>
<td></td>
</tr>
<tr>
<td>void OnSetOverwriteColor (const Ptr&lt; SetOverwriteColor &gt; &amp;msg)</td>
<td>on set overwrite color</td>
</tr>
<tr>
<td>void OnSetShaderVariable (const Ptr&lt; SetShaderVariable &gt; &amp;msg)</td>
<td>set shader variable</td>
</tr>
<tr>
<td>void SetEntity (const Ptr&lt; Entity &gt; &amp;v)</td>
<td>Set entity, this is attached to, to <code>v</code>.</td>
</tr>
<tr>
<td>void ClearEntity ()</td>
<td>Remove entity.</td>
</tr>
<tr>
<td>void RegisterMessage (const Id &amp;msgId)</td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void GraphicsFeature::GraphicsProperty::OnActivate ( ) [virtual]
called from Entity::ActivateProperties()
Attach the property to a game entity. This will create and setup the required graphics entities.
Reimplemented from Game::Property.
Reimplemented in StateObjectFeature::StateGraphicsProperty.
```

```cpp
void GraphicsFeature::GraphicsProperty::OnDeactivate ( ) [virtual]
called from Entity::DeactivateProperties()
Remove the property from its game entity. This will release the graphics entities owned by the property.
Reimplemented from Game::Property.
Reimplemented in StateObjectFeature::StateGraphicsProperty.
```

```cpp
const Util::String & GraphicsFeature::GraphicsProperty::GetGraphicsResource ( ) const [inline, virtual]
override to provide a self managed graphics resource (default is Attr::Graphics)
Get the default graphics resource, which is Attr::Graphics. subclasses may override this to provide a self managed resource.
```

```cpp
void GraphicsFeature::GraphicsProperty::SetupGraphicsEntities ( ) [protected, virtual]
setup graphics entities
```
Setup the graphics entities. You may override this method in a subclass if different setup is needed.

Reimplemented in `GraphicsFeature::ActorGraphicsProperty`.

```c++
void GraphicsFeature::GraphicsProperty::UpdateTransform(const Math::matrix44& m, bool setDirectly = false) [protected, virtual]
```

Update the graphics entity's transform

Called to update the graphics entity's transform.

```c++
void GraphicsFeature::GraphicsProperty::SetVisible(bool visible) [protected]
```

Shows or hides all attached graphics entities.

```c++
void Game::Property::OnLoad() [virtual, inherited]
```

called from within `Entity::Load()` after attributes are loaded

This method is called from within `Game::Entity::Load()` after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in `PhysicsFeature::TriggerProperty`, and `StateObjectFeature::StateProperty`.

```c++
void Game::Property::OnStart() [virtual, inherited]
```

called from within `Entity::OnStart()` after `OnLoad` when the complete world exists
This method is called from within `Game::Entity::OnStart()`. This is the moment when the world is complete and the entity can establish connections to other entities.

Reimplemented in `GraphicsFeature::CameraProperty`.

```cpp
void Game::Property::OnSave()
```

called from within `Entity::Save()` before attributes are saved back to database

This method is called from within `Game::Entity::Save()` before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in `PhysicsFeature::TriggerProperty`, and `StateObjectFeature::StateProperty`.

```cpp
void Game::Property::OnBeginFrame()
```

called on begin of frame

This method is called from `Game::Entity::OnBeginFrame()` on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in `PhysicsFeature::TriggerProperty`, and `StateObjectFeature::StateProperty`.

```cpp
void Game::Property::OnMoveBefore()
```

called before movement happens

This method is called from `Game::Entity::OnMoveBefore()` on all properties attached to an entity in the order of attachment. Override
this method if your property has any work to do before the physics subsystem is triggered.

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`.

```cpp
void Game::Property::OnMoveAfter() [virtual, inherited]
```
called after movement has happened

This method is called from `Game::Entity::OnMoveAfter()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`, and `PhysicsFeature::PhysicsProperty`.

```cpp
void Game::Property::OnRender() [virtual, inherited]
```
called before rendering happens

This method is called from `Game::Entity::OnRender()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before rendering happens.

Reimplemented in `GraphicsFeature::CameraProperty`, `GraphicsFeature::ChaseCameraProperty`, and `GraphicsFeature::MayaCameraProperty`.

```cpp
void Game::Property::OnLoseActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from `Game::Entity::OnLoseActivity()` on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active
camera).

Reimplemented in **PhysicsFeature::ActorPhysicsProperty**.

```cpp
void Game::Property::OnGainActivity( ) [virtual, inherited]
```
called when game debug visualization is on

This method is called from `Game::Entity::OnRenderDebug()` on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

```cpp
void Messaging::Port::AttachHandler (const Ptr<Handler> &h ) [inherited]
```
attach a message handler to the port

**Attach a message handler to the port.**

```cpp
void Messaging::Port::RemoveHandler (const Ptr<Handler> &h ) [inherited]
```
remove a message handler from the port

**Remove a message handler from the port.**

```cpp
void Messaging::Port::Send (const Ptr<Message> &msg ) [virtual, inherited]
```
send a message to the port

**Send a message to the port.** This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag
of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
GraphicsFeature::LightFlickerUtil
GraphicsFeature::LightFlickerUtil Class Reference

#include <lightflickerutil.h>
Detailed Description

Manipulate a light entity with position and intensity flickering.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LightFlickerUtil ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>SetLightEntity (const Ptr&lt; Graphics::AbstractLightEntity &gt;&amp; e)</strong></td>
<td>set light entity to manipulate</td>
</tr>
<tr>
<td><strong>GetLightEntity () const</strong></td>
<td>get light entity</td>
</tr>
<tr>
<td><strong>HasLightEntity () const</strong></td>
<td>return true if entity is valid</td>
</tr>
<tr>
<td><strong>setEnabled (bool b)</strong></td>
<td>enable/disable flickering</td>
</tr>
<tr>
<td><strong>IsEnabled () const</strong></td>
<td>is flickering enabled?</td>
</tr>
<tr>
<td><strong>SetFrequency (float f)</strong></td>
<td>set flicker frequency</td>
</tr>
<tr>
<td><strong>GetFrequency () const</strong></td>
<td>get flicker frequency</td>
</tr>
<tr>
<td><strong>SetIntensityAmplitude (float f)</strong></td>
<td>set intensity flicker amplitude</td>
</tr>
<tr>
<td><strong>GetIntensityAmplitude () const</strong></td>
<td>get intensity flicker amplitude</td>
</tr>
<tr>
<td><strong>setPositionAmplitude (float f)</strong></td>
<td>set position flicker amplitude</td>
</tr>
<tr>
<td><strong>getPositionAmplitude () const</strong></td>
<td>get position flicker amplitude</td>
</tr>
<tr>
<td><strong>Update ()</strong></td>
<td>update the light source, call this method once per frame</td>
</tr>
</tbody>
</table>
Member Function Documentation

void GraphicsFeature::LightFlickerUtil::Update ( )

update the light source, call this method once per frame

This changes the transformation offset matrix and the intensity modulation of the light entity.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

GraphicsFeature::MayaCameraProperty
GraphicsFeature::MayaCameraProperty
Class Reference

#include <mayacameraproperty.h>

Inheritance diagram for GraphicsFeature::MayaCameraProperty:
Detailed Description

A manually controlled camera property which implements different control models.

(C) 2008 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th><code>CallbackType</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>callback types</code></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MayaCameraProperty ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~MayaCameraProperty ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual void OnActivate ()</strong></td>
<td>called from Entity::ActivateProperties()</td>
</tr>
<tr>
<td><strong>virtual void OnDeactivate ()</strong></td>
<td>called from Entity::DeactivateProperties()</td>
</tr>
<tr>
<td><strong>virtual void SetupCallbacks ()</strong></td>
<td>setup callbacks for this property</td>
</tr>
<tr>
<td><strong>virtual void OnRender ()</strong></td>
<td>called before rendering happens</td>
</tr>
<tr>
<td><strong>virtual void SetupDefaultAttributes ()</strong></td>
<td>setup default entity attributes</td>
</tr>
<tr>
<td><strong>virtual void SetupAcceptedMessages ()</strong></td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td><strong>virtual void HandleMessage (const Ptr<a href="">Messaging::Message</a> &amp;msg)</strong></td>
<td>handle a single message</td>
</tr>
<tr>
<td><strong>virtual void OnObtainInputFocus ()</strong></td>
<td>called when input focus is gained</td>
</tr>
<tr>
<td><strong>virtual void OnLoseInputFocus ()</strong></td>
<td>called when input focus is lost</td>
</tr>
<tr>
<td><strong>virtual void OnStart ()</strong></td>
<td>called from within Entity::OnStart() after OnLoad when the complete world exist</td>
</tr>
<tr>
<td><strong>virtual void OnObtainCameraFocus ()</strong></td>
<td>called when camera focus is obtained</td>
</tr>
<tr>
<td><strong>virtual void OnLoseCameraFocus ()</strong></td>
<td>called when camera focus is lost</td>
</tr>
<tr>
<td><strong>virtual bool HasFocus () const</strong></td>
<td>return true if currently has camera focus</td>
</tr>
<tr>
<td><strong>const Ptr&lt;Entity&gt; &amp; GetEntity () const</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>HasEntity</strong> () const</td>
<td>return true if entity pointer is valid</td>
</tr>
<tr>
<td><strong>IsActive</strong> () const</td>
<td>return true if property is currently active</td>
</tr>
<tr>
<td><strong>OnLoad</strong> ()</td>
<td>called from within Entity::Load() after attributes are loaded</td>
</tr>
<tr>
<td><strong>OnSave</strong> ()</td>
<td>called from within Entity::Save() before attributes are saved back to database</td>
</tr>
<tr>
<td><strong>OnBeginFrame</strong> ()</td>
<td>called on begin of frame</td>
</tr>
<tr>
<td><strong>OnMoveBefore</strong> ()</td>
<td>called before movement happens</td>
</tr>
<tr>
<td><strong>OnMoveAfter</strong> ()</td>
<td>called after movement has happened</td>
</tr>
<tr>
<td><strong>OnRenderDebug</strong> ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td><strong>OnLoseActivity</strong> ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td><strong>OnGainActivity</strong> ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td><strong>AttachHandler</strong> (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td><strong>RemoveHandler</strong> (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td><strong>RemoveAllHandlers</strong> ()</td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td><strong>GetNumHandlers</strong> () const</td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td>const Ptr&lt; Handler &gt; &amp;</td>
<td><strong>GetHandlerAtIndex</strong> (IndexT i) const</td>
</tr>
<tr>
<td><strong>Send</strong> (const Ptr&lt; Message &gt; &amp;msg)</td>
<td>send a message to the port</td>
</tr>
<tr>
<td>const Util::Array&lt; const Id * &gt; &amp;</td>
<td><strong>GetAcceptedMessages</strong> () const</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>AcceptsMessage</code> (const Id &amp;msgId) const</td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td><code>GetRefCount</code> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef</code> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release</code> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf</code> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf</code> (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf</code> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA</code> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA</code> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA</code> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName</code> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC</code> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
**Protected Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>UpdateAudioListenerPosition()</code> const</td>
<td>update audio listener position</td>
</tr>
<tr>
<td><code>SetEntity(const Ptr&lt; Entity &gt;&amp; v)</code></td>
<td>Set entity, this is attached to, to <code>v</code>.</td>
</tr>
<tr>
<td><code>ClearEntity()</code></td>
<td>Remove entity.</td>
</tr>
<tr>
<td><code>RegisterMessage(const Id &amp;msgId)</code></td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

void GraphicsFeature::MayaCameraProperty::SetupDefaultAttributes( ) [virtual]

setup default entity attributes

This adds the default attributes to the property.

Reimplemented from GraphicsFeature::CameraProperty.

void GraphicsFeature::CameraProperty::OnObtainCameraFocus( ) [virtual, inherited]

called when camera focus is obtained

This method is called by the FocusManager when our entity gains the camera focus. Override this method if your subclass needs to do some initialization when gaining the camera focus.

Reimplemented in GraphicsFeature::ChaseCameraProperty.

void GraphicsFeature::CameraProperty::OnLoseCameraFocus( ) [virtual, inherited]

called when camera focus is lost

This method is called by the FocusManager when our entity loses the camera focus. Override this method if your subclass needs to do any cleanup work here.

bool GraphicsFeature::CameraProperty::HasFocus( ) const [virtual, inherited]

return true if currently has camera focus

This method returns true if our entity has the camera focus. This implementation makes sure that 2 properties cannot report that they have the camera focus by accident.
void Game::Property::OnLoad() [virtual, inherited]
called from within Entity::Load() after attributes are loaded

This method is called from within Game::Entity::Load() after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void Game::Property::OnSave() [virtual, inherited]
called from within Entity::Save() before attributes are saved back to database

This method is called from within Game::Entity::Save() before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void Game::Property::OnBeginFrame() [virtual, inherited]
called on begin of frame

This method is called from Game::Entity::OnBeginFrame() on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in PhysicsFeature::TriggerProperty, and
StateObjectFeature::StateProperty.

void
Game::Property::OnMoveBefore( ) [virtual, inherited]
called before movement happens

This method is called from Game::Entity::OnMoveBefore() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before the physics subsystem is triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty.

void
Game::Property::OnMoveAfter( ) [virtual, inherited]
called after movement has happened

This method is called from Game::Entity::OnMoveAfter() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty, and PhysicsFeature::PhysicsProperty.

void
Game::Property::OnRenderDebug( ) [virtual, inherited]
called when game debug visualization is on

This method is called from Game::Entity::OnRenderDebug() on all properties attached to an entity in the order of attachment. It's meant for debug issues. It will be called when debug mode is enabled.

Reimplemented in GraphicsFeature::GraphicsProperty, PhysicsFeature::ActorPhysicsProperty, and PhysicsFeature::TriggerProperty.

void
Game::Property::OnLoseActivity( ) [virtual, inherited]
called when game debug visualization is on

This method is called from Game::Entity::OnLoseActivity() on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

Reimplemented in PhysicsFeature::ActorPhysicsProperty.

```cpp
void Game::Property::OnGainActivity()
```

called when game debug visualization is on

This method is called from Game::Entity::OnRenderDebug() on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

```cpp
void Messaging::Port::AttachHandler(const Ptr<Handler> &h)
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler(const Ptr<Handler> &h)
```

remove a message handler from the port

Remove a message handler from the port.

```cpp
void Messaging::Port::Send(const Ptr<Message> &msg)
```
send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
GraphicsFeature::SegmentedGfxUtil
GraphicsFeature::SegmentedGfxUtil
Class Reference

#include <segmentedgfxutil.h>
Detailed Description

Check if a Nebula3 graphics resource consists of hierarchy nodes below the toplevel node, and if yes, create one graphics entity for each hierarchy node. Otherwise create just a single graphics entity. This segmentation helps in visibility culling large environmental objects.

(C) 2007 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SegmentedGfxUtil()</code></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td><code>Util::Array&lt; Ptr&lt; Graphics::ModelEntity &gt; &gt;</code></td>
<td><code>CreateAndSetupGraphicsEntities</code></td>
</tr>
<tr>
<td></td>
<td>(const <code>Util::String</code> &amp;resName, const <code>Math::matrix44</code> &amp;worldMatrix, const <code>Ptr&lt; Graphics::Stage&gt;</code> &amp;stage=0)</td>
</tr>
<tr>
<td></td>
<td><em>create and setup one or more graphics entities from hierarchy nodes</em></td>
</tr>
</tbody>
</table>
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Http::DefaultHttpRequestHandler
Http::DefaultHttpRequestHandler Class Reference

#include <defaulthttprequesthandler.h>

Inheritance diagram for Http::DefaultHttpRequestHandler:
Detailed Description

This implements the "Home Page" of the Nebula3 application. It will answer all HTTP requests which are not handled by a custom HttpRequestHandler.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual void HandleRequest (const Ptr&lt; HttpRequest &gt; &amp;request)</code></td>
<td>Handle a HTTP request, the handler is expected to fill the content stream with response data.</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetName () const</code></td>
<td>Get a human readable name of the request handler.</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetDesc () const</code></td>
<td>Get a human readable description of the request handler.</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetRootLocation () const</code></td>
<td>Get a resource location path which is accepted by the handler (e.g. <code>/display</code>).</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>Get the current refcount.</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>Increment refcount by one.</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>Decrement refcount and destroy object if refcount is zero.</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class.</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>Return true if this object is instance of given class by string.</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by fourcc.</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class.</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string.</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc.</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>Get the class name.</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HandlePendingRequests()</code></td>
<td>handle all pending requests, called by local-thread's <code>HttpServerProxy</code></td>
</tr>
<tr>
<td><code>PutRequest(const Ptr&lt; HttpRequest &gt; &amp;httpRequest)</code></td>
<td>put a request to the pending queue, called by <code>HttpServer</code> thread</td>
</tr>
<tr>
<td><code>SetName(const Util::String &amp;n)</code></td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><code>SetDesc(const Util::String &amp;d)</code></td>
<td>set human readable description</td>
</tr>
<tr>
<td><code>SetRootLocation(const Util::String &amp;l)</code></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Http::HttpRequestHandler::HandlePendingRequests() [protected, inherited]

handle all pending requests, called by local-thread's HttpServerProxy

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

void
Http::HttpRequestHandler::PutRequest(const Ptr<HttpRequest> &httpRequest) [protected, inherited]

put a request to the pending queue, called by HttpServer thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

int
Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const **Util::String** &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

ger the class name

Get the class name of the object.

**Util::FourCC**
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

ger the class FourCC code

Get the class FourCC of the object.

**void**
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HtmlElement
Http::HtmlElement Class Reference

#include <html4element.h>
Detailed Description

HTML markup elements.

(C) 2007 Radon Labs GmbH
Public Types

enum Code
  elements
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static Util::String</th>
<th>ToHtml (Code c)</th>
</tr>
</thead>
</table>

*convert to string*
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HtmlPageWriter
Http::HtmlPageWriter Class Reference

#include <htmlpagewriter.h>

Inheritance diagram for Http::HtmlPageWriter:
Detailed Description

A stream writer which supports writing a HTML-formatted page into a stream.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HtmlPageWriter()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~HtmlPageWriter()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>SetTitle()</code></td>
<td>set the title of the page</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>begin writing the stream</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>end writing the stream</td>
</tr>
<tr>
<td><code>AddAttr()</code></td>
<td>add an attribute for the next element</td>
</tr>
<tr>
<td><code>Begin()</code></td>
<td>begin a generic element</td>
</tr>
<tr>
<td><code>End()</code></td>
<td>end a generic element</td>
</tr>
<tr>
<td><code>Element()</code></td>
<td>shortcut for <code>Begin()/Text()/End()</code></td>
</tr>
<tr>
<td><code>LineBreak()</code></td>
<td>write a line break</td>
</tr>
<tr>
<td><code>HorizontalRule()</code></td>
<td>write a horizontal rule</td>
</tr>
<tr>
<td><code>Text()</code></td>
<td>add inline text</td>
</tr>
<tr>
<td><code>TableRow2()</code></td>
<td>write a 2-element table row</td>
</tr>
<tr>
<td><code>SetStream()</code></td>
<td>set stream to write to</td>
</tr>
<tr>
<td><code>GetStream()</code></td>
<td>get currently set stream</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool HasStream () const</code></td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td><code>bool IsOpen () const</code></td>
<td>return true if currently open</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
add an attribute for the next element

Adds an attribute for the next HTML element. Any number of attributes can be defined before calling Begin(). The attribute stack will be cleared as Begin() returns.

begin a generic element

Begin a generic HTML element. Elements can be nested.

down an attribute for the next element

Adds an attribute for the next HTML element. Any number of attributes can be defined before calling Begin(). The attribute stack will be cleared as Begin() returns.

end a generic element

End a generic HTML element. The element must match the last Begin() call!

set stream to write to

Attaches the writer to a stream. This will increment the refcount of the stream.
Reimplemented in **Messaging::MessageWriter**.

```cpp
const Ptr< Stream > &
IO::StreamWriter::GetStream( ) const [inherited]

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use **HasStream()** to determine if a stream is attached.

```cpp
bool
IO::StreamWriter::HasStream( ) const [inherited]

return true if a stream is set

Returns true if a stream is attached to the writer.

```cpp
int
Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void
Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]
get the class name

Get the class name of the object.

**Util::FourCC**
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

**void**
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
HttpClient
Http::HttpClient Class Reference

#include <httpclient.h>
Detailed Description

Use a HTTP client to send HTTP requests to a HTTP server, and receive and decode HTTP responses. The `HttpClient` class is generally blocking. For non-blocking behaviour it's best to wrap the `HttpClient` into a thread.

(C) 2009 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpClientRegistry
Http::HttpClientRegistry Class Reference

#include <httpclientregistry.h>
Detailed Description

The **HttpClientRegistry** provides a way to re-use existing connections to HTTP servers instead of setting up a HTTP connection for every single HTTP request.

(C) 2009 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpInterface
Http::HttpInterface Class Reference

#include <httpinterface.h>

Inheritance diagram for Http::HttpInterface:

- Core::RefCounted
- Messaging::AsyncPort
- Interface::InterfaceBase
- Http::HttpInterface
Detailed Description

The **HttpInterface** launches the **HttpServer** thread and is the communication interface with the **HttpServer** thread.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HttpInterface ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~HttpInterface ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>Open ()</strong></td>
<td>Open the interface object</td>
</tr>
<tr>
<td>virtual void <strong>AttachHandler</strong> (const <strong>Ptr</strong>&lt; <strong>Messaging::Handler</strong> &gt; &amp;h)</td>
<td>Attach a handler to the port (call before open!)</td>
</tr>
<tr>
<td>const <strong>Util::StringAtom &amp;</strong> <strong>GetCompanyName ()</strong> const</td>
<td>Get the company name</td>
</tr>
<tr>
<td>const <strong>Util::StringAtom &amp;</strong> <strong>GetAppName ()</strong> const</td>
<td>Get the application name</td>
</tr>
<tr>
<td>const <strong>Util::StringAtom &amp;</strong> <strong>GetRootDirectory ()</strong> const</td>
<td>Get the root directory</td>
</tr>
<tr>
<td>void <strong>SetHandlerThread</strong> (const <strong>Ptr</strong>&lt; <strong>HandlerThreadBase</strong> &gt; &amp;handlerThread)</td>
<td>Set pointer to handler thread object (must be derived from <strong>HandlerThreadBase</strong>)</td>
</tr>
<tr>
<td>const <strong>Ptr</strong>&lt; <strong>HandlerThreadBase</strong> &gt; &amp; <strong>GetHandlerThread ()</strong> const</td>
<td>Get pointer to handler thread object</td>
</tr>
<tr>
<td>virtual void <strong>RemoveHandler</strong> (const <strong>Ptr</strong>&lt; <strong>Handler</strong> &gt; &amp;h)</td>
<td>Dynamically remove a handler from the port</td>
</tr>
<tr>
<td>virtual void <strong>Close ()</strong></td>
<td>Close the async port</td>
</tr>
<tr>
<td>bool <strong>IsOpen ()</strong> const</td>
<td>Return true if port is open</td>
</tr>
<tr>
<td>template&lt;class MESSAGETYPE&gt; void <strong>Send</strong> (const <strong>Ptr</strong>&lt; MESSAGETYPE &gt; &amp;msg)</td>
<td>Send an asynchronous message to the port</td>
</tr>
<tr>
<td>template&lt;class MESSAGETYPE&gt; void <strong>SendWait</strong> (const <strong>Ptr</strong>&lt; MESSAGETYPE &gt; &amp;msg)</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>void Wait (const Ptr&lt;MESSAGETYPE&gt; &amp;msg)</td>
<td>wait for a message to be handled</td>
</tr>
<tr>
<td>bool Peek (const Ptr&lt;MESSAGETYPE&gt; &amp;msg)</td>
<td>peek a message whether it has been handled</td>
</tr>
<tr>
<td>void Cancel (const Ptr&lt;MESSAGETYPE&gt; &amp;msg)</td>
<td>cancel a pending message</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC &amp; GetClassFourCC () const</td>
<td>get the class name</td>
</tr>
</tbody>
</table>
get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void Messaging::AsyncPort::RemoveHandler(const Ptr<Handler> &h) [virtual, inherited]
```

dynamically remove a handler from the port

Dynamically remove a message handler.

```cpp
void Messaging::AsyncPort::Close() [virtual, inherited]
```

close the async port

Closes the async port.

Reimplemented in **Graphics::GraphicsInterface**.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const `Util::String` & Core::RefCounted::GetClassName

get the class name

Get the class name of the object.

`Util::FourCC` Core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

`void` Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpMessageHandler
#include <httpmessagehandler.h>

Inheritance diagram for Http::HttpMessageHandler:
Detailed Description

Runs the **HttpServer** thread, and owns the central http server. Processes messages sent to the **HttpInterface** from other threads.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HttpMessageHandler ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~HttpMessageHandler ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <strong>Open ()</strong></td>
<td>open the handler</td>
</tr>
<tr>
<td>virtual void <strong>Close ()</strong></td>
<td>close the handler</td>
</tr>
<tr>
<td>virtual bool <strong>HandleMessage</strong> (const <strong>Ptr</strong>&lt;Messaging::Message &gt; &amp;msg)</td>
<td>handle a message, return true if handled</td>
</tr>
<tr>
<td>virtual void <strong>DoWork ()</strong></td>
<td>do per-frame work</td>
</tr>
<tr>
<td><strong>SetCompanyName</strong> (const Util::StringAtom &amp;companyName)</td>
<td>set the company name</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; <strong>GetCompanyName ()</strong> const</td>
<td>get the company name</td>
</tr>
<tr>
<td>void <strong>SetAppName</strong> (const Util::StringAtom &amp;appName)</td>
<td>set the application name</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; <strong>GetAppName ()</strong> const</td>
<td>get the application name</td>
</tr>
<tr>
<td>bool <strong>IsOpen ()</strong> const</td>
<td>return true if open</td>
</tr>
<tr>
<td>int <strong>GetRefCount ()</strong> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DumpRefCountingLeaks()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void
Http::HttpMessageHandler::DoWork( ) [virtual]
```
do per-frame work
Triggers the http server from time to time.
Reimplemented from Interface::InterfaceHandlerBase.

```cpp
int
Core::RefCounted::GetRefCount( ) const [inline, inherited]
```
get the current refcount
Return the current refcount of the object.

```cpp
void
Core::RefCounted::AddRef( ) [inline, inherited]
```
increment refcount by one
Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release( ) [inline, inherited]
```
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]
```
get the class name
Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpMethod
Http::HttpMethod Class Reference

#include <httpmethod.h>
Detailed Description

Http methods enumeration (GET, PUT, etc...).

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>http methods</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <strong>Code</strong></td>
<td><strong>FromString</strong> (const <strong>Util::String</strong> &amp;str) convert from string</td>
</tr>
<tr>
<td>static <strong>Util::String</strong></td>
<td><strong>ToString</strong> (<strong>Code</strong> c) convert to string</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpNzStream
Http::HttpNzStream Class Reference

#include <httpnzstream.h>
Detailed Description

A version of **HttpStream** which reads .nz compressed files as created by the Nebula3 archiver tool. .nz is a simple container for ZLIB compressed data.

(C) 2009 Radon Labs GmbH
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**HttpRequest**
#include <httprequest.h>

Inheritance diagram for Http::HttpRequest:
Detailed Description

Encapsulates a complete Http request into a message.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HttpRequest ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~HttpRequest ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <strong>SetMethod (HttpMethod::Code requestMethod)</strong></td>
<td>set the <em>Http</em> method (GET, PUT, etc...)</td>
</tr>
<tr>
<td><strong>HttpMethod::Code GetMethod () const</strong></td>
<td>get the <em>Http</em> method</td>
</tr>
<tr>
<td>void <strong>SetURI (const IO::URI &amp;requestUri)</strong></td>
<td>set the request URI</td>
</tr>
<tr>
<td><strong>const IO::URI &amp; GetURI () const</strong></td>
<td>get the request URI</td>
</tr>
<tr>
<td>void <strong>SetResponseContentStream (const Ptr<a href="">IO::Stream</a> &amp;responseContentStream)</strong></td>
<td>set the response content stream</td>
</tr>
<tr>
<td><strong>const Ptr<a href="">IO::Stream</a> &amp; GetResponseContentStream () const</strong></td>
<td>get the response content stream</td>
</tr>
<tr>
<td>void <strong>SetStatus (HttpStatus::Code status)</strong></td>
<td>set the http status (set by <em>HttpRequestHandler</em>)</td>
</tr>
<tr>
<td><strong>HttpStatus::Code GetStatus () const</strong></td>
<td>get the http status</td>
</tr>
<tr>
<td>bool <strong>CheckId (const Messaging::Id &amp;id) const</strong></td>
<td>return true if message is of the given id</td>
</tr>
<tr>
<td>virtual void <strong>Encode (const Ptr<a href="">IO::BinaryWriter</a> &amp;writer)</strong></td>
<td>encode message into a stream</td>
</tr>
<tr>
<td>virtual void <strong>Decode (const Ptr<a href="">IO::BinaryReader</a> &amp;reader)</strong></td>
<td>decode message from a stream</td>
</tr>
<tr>
<td>void <strong>SetHandled (bool b)</strong></td>
<td>set the handled flag</td>
</tr>
<tr>
<td>bool <strong>Handled () const</strong></td>
<td></td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>SetDeferred (bool b)</code></td>
<td>set deferred flag</td>
</tr>
<tr>
<td><code>IsDeferred () const</code></td>
<td>get deferred flag</td>
</tr>
<tr>
<td><code>SetDeferredHandled (bool b)</code></td>
<td>set the deferred handled flag</td>
</tr>
<tr>
<td><code>DeferredHandled () const</code></td>
<td>get the deferred handled flag</td>
</tr>
<tr>
<td><code>GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpRequestHandler
Http::HttpRequestHandler Class Reference

#include <httprequesthandler.h>

Inheritance diagram for Http::HttpRequestHandler:
HttpRequestHandlers are attached to the HttpServer and process incoming HTTP requests. When an Http request comes in, the HttpServer asks every attached HttpRequestHandler until the first one accepts the request. If the HttpRequestHandler accepts the request its HandleRequest() method will be called with a pointer to a content stream. The request handler is expected to write the response to the content stream (IMPORTANT: don't forget to set the MediaType on the stream!) and return with a HttpStatus code.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HttpRequestHandler ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>virtual ~HttpRequestHandler ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetName () const</code></td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetDesc () const</code></td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetRootLocation () const</code></td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/display&quot;)</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool Isa (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool Isa (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool Isa (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td><code>Util::FourCC</code> <strong>GetClassFourCC</strong> () const</td>
<td></td>
</tr>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void HandleRequest (const Ptr&lt; HttpRequest &gt; &amp;request)</td>
<td>handle a http request, overwrite this method in your subclass</td>
</tr>
<tr>
<td>void HandlePendingRequests ()</td>
<td>handle all pending requests, called by local-thread's HttpServerProxy</td>
</tr>
<tr>
<td>void PutRequest (const Ptr&lt; HttpRequest &gt; &amp;httpRequest)</td>
<td>put a request to the pending queue, called by HttpServer thread</td>
</tr>
<tr>
<td>void SetName (const Util::String &amp;n)</td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td>void SetDesc (const Util::String &amp;d)</td>
<td>set human readable description</td>
</tr>
<tr>
<td>void SetRootLocation (const Util::String &amp;l)</td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void Http::HttpRequestHandler::HandleRequest (const Ptr<HttpRequest> & request) [protected, virtual]
```

handle a http request, overwrite this method in your subclass

Overwrite this method in your subclass. This method will be called by the `HttpServer` if `AcceptsRequest()` returned true. The request handler should properly process the request by filling the `responseContentStream` with data (for instance a HTML page), set the `MediaType` on the `responseContentStream` (for instance "text/html") and return with a `HttpStatus` code (usually `HttpStatus::OK`).

Reimplemented in `Debug::CorePageHandler`, `Debug::DebugPageHandler`, `Debug::HelloWorldRequestHandler`, `Debug::SvgTestPageHandler`, `Http::DefaultHttpRequestHandler`, `Debug::ConsolePageHandler`, `Debug::IoPageHandler`, `Debug::MemoryPageHandler`, `Debug::ThreadPageHandler`, `Debug::DisplayPageHandler`, `Debug::MeshPageHandler`, `Debug::ShaderPageHandler`, `Debug::TexturePageHandler`, `Debug::GraphicsPageHandler`, `Debug::StreamingTexturePageHandler`, and `Debug::ObjectInspectorHandler`.

```cpp
void Http::HttpRequestHandler::PutRequest (const Ptr<HttpRequest> & httpRequest) [protected]
```

handle all pending requests, called by local-thread's `HttpServerProxy`

Handle all pending http requests in the pending queue. This method must be called frequently from the thread which created this request handler.

```cpp
void Http::HttpRequestHandler::HandlePendingRequests () [protected]
```
Put a request to the pending queue, called by **HttpServer** thread

Put a http request into the request handlers message queue. This method is meant to be called from another thread.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpRequestReader
Http::HttpRequestReader Class Reference

#include <httprequestreader.h>

Inheritance diagram for Http::HttpRequestReader:
Detailed Description

A stream reader which cracks a HTTP request into its components.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HttpRequestReader ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>bool ReadRequest ()</code></td>
<td>decode the request from the stream (call first before Get methods!)</td>
</tr>
<tr>
<td><code>bool IsValidHttpRequest () const</code></td>
<td>return true if the stream contains a valid HTTP request</td>
</tr>
<tr>
<td><code>HttpMethod::Code GetHttpMethod () const</code></td>
<td>get HTTP request method</td>
</tr>
<tr>
<td><code>const IO::URI &amp; GetRequestURI () const</code></td>
<td>get request URI</td>
</tr>
<tr>
<td><code>void SetStream (const Ptr&lt; Stream &gt; &amp;s)</code></td>
<td>set stream to read from</td>
</tr>
<tr>
<td><code>const Ptr&lt; Stream &gt; &amp; GetStream () const</code></td>
<td>get currently set stream</td>
</tr>
<tr>
<td><code>bool HasStream () const</code></td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td><code>bool Eof () const</code></td>
<td>return true if the stream has reached EOF</td>
</tr>
<tr>
<td><code>virtual bool Open ()</code></td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td><code>virtual void Close ()</code></td>
<td>end reading from the stream</td>
</tr>
<tr>
<td><code>bool IsOpen () const</code></td>
<td>return true if currently open</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>Util::String</strong> &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void IO::StreamReader::SetStream(const Ptr<Stream> & s) [inherited]
```

set stream to read from

Attaches the reader to a stream. This will increment the refcount of the stream.

Reimplemented in `Messaging::MessageReader`.

```cpp
const Ptr<Stream> & IO::StreamReader::GetStream() const [inherited]
```

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use `HasStream()` to determine if a stream is attached.

```cpp
bool IO::StreamReader::HasStream() const [inherited]
```

return true if a stream is set

Returns true if a stream is attached to the reader.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Http::HttpRequestWriter
Http::HttpRequestWriter Class Reference

#include <httprequestwriter.h>

Inheritance diagram for Http::HttpRequestWriter:
Detailed Description

 Writes a valid HTTP request header to a stream.

 (C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HttpRequestWriter()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>void SetMethod(HttpMethod::Code httpMethod)</code></td>
<td>set HTTP Method</td>
</tr>
<tr>
<td><code>void SetURI(const IO::URI &amp;uri)</code></td>
<td>set the URI of the request</td>
</tr>
<tr>
<td><code>void SetUserAgent(const Util::String &amp;userAgent)</code></td>
<td>set optional User-Agent string</td>
</tr>
<tr>
<td><code>bool WriteRequestHeader()</code></td>
<td>write the request header to the stream</td>
</tr>
<tr>
<td><code>void SetStream(const Ptr&lt;Stream&gt; &amp;s)</code></td>
<td>set stream to write to</td>
</tr>
<tr>
<td><code>const Ptr&lt;Stream&gt; &amp; GetStream()</code></td>
<td>get currently set stream</td>
</tr>
<tr>
<td><code>bool HasStream()</code></td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td><code>virtual bool Open()</code></td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td><code>virtual void Close()</code></td>
<td>end reading from the stream</td>
</tr>
<tr>
<td><code>bool isOpen()</code></td>
<td>return true if currently open</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only)!</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```c++
void IO::StreamWriter::SetStream(const Ptr<Stream> & s) [inherited]
```

set stream to write to

Attaches the writer to a stream. This will increment the refcount of the stream.

Reimplemented in `Messaging::MessageWriter`.

```c++
const Ptr<Stream> & IO::StreamWriter::GetStream () const [inherited]
```

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use `HasStream()` to determine if a stream is attached.

```c++
bool IO::StreamWriter::HasStream () const [inherited]
```

return true if a stream is set

Returns true if a stream is attached to the writer.

```c++
int Core::RefCounted::GetRefCount () const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```c++
void Core::RefCounted::AddRef () [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Http::HttpResponseReader
#include <httpresponsereader.h>

Inheritance diagram for Http::HttpResponseReader:
Detailed Description

Decodes a response header from a HTTP server and optionally writes received content data to a provided stream.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HttpResponseReader ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>bool <strong>ReadResponse ()</strong></td>
<td>read the response</td>
</tr>
<tr>
<td>bool <strong>IsValidHttpResponse ()</strong> const</td>
<td>return true if this was a valid response</td>
</tr>
<tr>
<td><strong>HttpStatus::Code</strong> GetStatusCode () const</td>
<td>get the HTTP status code which was sent by the server</td>
</tr>
<tr>
<td>const <strong>IO::MediaType &amp;</strong> GetContentType () const</td>
<td>get content type</td>
</tr>
<tr>
<td>SizeT <strong>GetContentLength ()</strong> const</td>
<td>get content length</td>
</tr>
<tr>
<td>void <strong>SetStream</strong> (const **Ptr&lt; Stream &gt; &amp;**s)</td>
<td>set stream to read from</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; Stream &gt; &amp;</strong> GetStream () const</td>
<td>get currently set stream</td>
</tr>
<tr>
<td>bool <strong>HasStream</strong> () const</td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td>bool <strong>Eof</strong> () const</td>
<td>return true if the stream has reached EOF</td>
</tr>
<tr>
<td>virtual bool <strong>Open</strong> ()</td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td>virtual void <strong>Close</strong> ()</td>
<td>end reading from the stream</td>
</tr>
<tr>
<td>bool <strong>IsOpen</strong> () const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const **Rtti &amp;**rtti) const</td>
<td>IsInstanceOf (const **Rtti &amp;**rtti) const</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td><code>return true if this object is instance of given class</code></td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td><code>return true if this object is instance of given class by string</code></td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td><code>return true if this object is instance of given class by fourcc</code></td>
</tr>
<tr>
<td><code>IsA</code></td>
<td><code>return true if this object is instance of given class, or a derived class</code></td>
</tr>
<tr>
<td><code>IsA</code></td>
<td><code>return true if this object is instance of given class, or a derived class, by string</code></td>
</tr>
<tr>
<td><code>IsA</code></td>
<td><code>return true if this object is instance of given class, or a derived class, by fourcc</code></td>
</tr>
<tr>
<td><code>GetClassName</code></td>
<td><code>get the class name</code></td>
</tr>
<tr>
<td><code>GetClassFourCC</code></td>
<td><code>get the class FourCC code</code></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>


Member Function Documentation

```cpp
void IO::StreamReader::SetStream(const Ptr<Stream> & s) [inherited]
```

set stream to read from

Attaches the reader to a stream. This will increment the refcount of the stream.

Reimplemented in Messaging::MessageReader.

```cpp
const Ptr<Stream> & IO::StreamReader::GetStream() const [inherited]
```

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use HasStream() to determine if a stream is attached.

```cpp
bool IO::StreamReader::HasStream() const [inherited]
```

return true if a stream is set

Returns true if a stream is attached to the reader.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```c
void
Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```c
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```c
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```c
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpResponseWriter
Http::HttpResponseWriter Class Reference

#include <httpresponsewriter.h>

Inheritance diagram for Http::HttpResponseWriter:
Detailed Description

Stream writer which writes a correct HTTP response to a stream.

(C) 2007 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>SetStatusCode</code> (HttpStatus::Code statusCode)</td>
<td>set status code</td>
</tr>
<tr>
<td>void <code>SetContent</code> (const Ptr<a href="">IO::Stream</a> &amp;contentStream)</td>
<td>set optional content stream (needs valid media type!)</td>
</tr>
<tr>
<td>void <code>WriteResponse</code> ()</td>
<td>write http response to the stream</td>
</tr>
<tr>
<td>void <code>SetStream</code> (const Ptr&lt;Stream&gt; &amp;s)</td>
<td>set stream to write to</td>
</tr>
<tr>
<td>const <code>GetStream</code> () const</td>
<td>get currently set stream</td>
</tr>
<tr>
<td>bool <code>HasStream</code> () const</td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td>virtual bool <code>Open</code> ()</td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td>virtual void <code>Close</code> ()</td>
<td>end reading from the stream</td>
</tr>
<tr>
<td>bool <code>IsOpen</code> () const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td>int <code>GetRefCount</code> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef</code> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release</code> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td></td>
<td><code>return true if this object is instance of given class, or a derived class</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><code>IsA (const Util::String &amp;rttiName) const</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>return true if this object is instance of given class, or a derived class, by string</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>return true if this object is instance of given class, or a derived class, by fourcc</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th><code>GetClassName () const</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>get the class name</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><code>GetClassFourCC () const</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>get the class FourCC code</code></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
void IO::StreamWriter::SetStream(const Ptr<Stream> & s) [inherited]

set stream to write to

Attaches the writer to a stream. This will increment the refcount of the stream.

Reimplemented in Messaging::MessageWriter.

const Ptr<Stream> & IO::StreamWriter::GetStream() const [inherited]

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use HasStream() to determine if a stream is attached.

bool IO::StreamWriter::HasStream() const [inherited]

return true if a stream is set

Returns true if a stream is attached to the writer.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpServer
Http::HttpServer Class Reference

#include <httpserver.h>

Inheritance diagram for Http::HttpServer:
Detailed Description

Implements an extremely simple standalone HTTP server with attached HttpRequestHandlers. Can be used to serve debug information about the Nebula3 application to web browsers.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HttpServer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~HttpServer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void SetPort (ushort p)</strong></td>
<td>set port number for http service</td>
</tr>
<tr>
<td><strong>ushort GetPort () const</strong></td>
<td>get port number of http service</td>
</tr>
<tr>
<td><strong>void SetSingleThreadMode (bool b)</strong></td>
<td>turn single-thread mode on/off (useful for debugging), default is off</td>
</tr>
<tr>
<td><strong>bool IsSingleThreadMode () const</strong></td>
<td>get single-thread mode</td>
</tr>
<tr>
<td><strong>bool Open ()</strong></td>
<td>open the http server</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close the http server</td>
</tr>
<tr>
<td><strong>bool IsOpen () const</strong></td>
<td>return true if server is open</td>
</tr>
<tr>
<td><strong>void AttachRequestHandler (const Ptr&lt; HttpRequestHandler &gt; &amp;h)</strong></td>
<td>attach a request handler to the server</td>
</tr>
<tr>
<td><strong>void RemoveRequestHandler (const Ptr&lt; HttpRequestHandler &gt; &amp;h)</strong></td>
<td>remove a request handler from the server</td>
</tr>
<tr>
<td><strong>Util::Array&lt; Ptr&lt; HttpRequestHandler &gt; &gt; GetRequestHandlers () const</strong></td>
<td>get registered request handlers</td>
</tr>
<tr>
<td><strong>void OnFrame ()</strong></td>
<td>call this method frequently to serve http connections</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```c++
void Http::HttpServer::SetSingleThreadMode ( bool b ) [inline]
```

turn single-thread mode on/off (useful for debugging), default is off

Switch on single thread mode, default is off. In single thread mode,
http requests will be processed immediately in the `OnFrame()`
method, not added to the request handler for asynchronous
processing. This may be useful for debugging, but is
dangerous/impossible if HTTP request handlers live in different
threads!!

```c++
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```c++
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```c++
void Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```c++
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

get the class name
Get the class name of the object.

`Util::FourCC` Core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

`void` Core::RefCounted::DumpRefCountingLeaks

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
<table>
<thead>
<tr>
<th>Main Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespaces</td>
</tr>
<tr>
<td>Data Structures</td>
</tr>
<tr>
<td>Files</td>
</tr>
<tr>
<td>Related Pages</td>
</tr>
<tr>
<td>Alphabetical List</td>
</tr>
<tr>
<td>Data Structures</td>
</tr>
<tr>
<td>Class Hierarchy</td>
</tr>
<tr>
<td>Data Fields</td>
</tr>
</tbody>
</table>

**Http::HttpServerProxy**
Http::HttpServerProxy Class Reference

#include <httpserverproxy.h>

Inheritance diagram for Http::HttpServerProxy:

```
Core::RefCounted
    \----> Http::HttpServerProxy
```
Detailed Description

Client-side proxy of the **HttpServer**. Client threads create and attach HttpRequestHandlers to their **HttpServerProxy**. The **HttpServerProxy** receives incoming http requests from the http thread, and lets its HttpRequestHandlers process the request in the client thread's context, then sends the result back to the http thread.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HttpServerProxy ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~HttpServerProxy ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>void Open ()</code></td>
<td>Open the server proxy</td>
</tr>
<tr>
<td><code>void Close ()</code></td>
<td>Close the server proxy</td>
</tr>
<tr>
<td><code>bool IsOpen () const</code></td>
<td>Return true if open</td>
</tr>
<tr>
<td><code>void AttachRequestHandler (const </code>Ptr<a href="Http::HttpRequestHandler">Http::HttpRequestHandler</a><code> &amp;h)</code></td>
<td>Attach a request handler to the server</td>
</tr>
<tr>
<td><code>void RemoveRequestHandler (const </code>Ptr<a href="Http::HttpRequestHandler">Http::HttpRequestHandler</a><code> &amp;h)</code></td>
<td>Remove a request handler from the server</td>
</tr>
<tr>
<td><code>void HandlePendingRequests ()</code></td>
<td>Handle pending HTTP requests, call this method frequently</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td></td>
</tr>
</tbody>
</table>
return true if this object is instance of given class, or a derived class

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
```
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpStatus
Http::HttpStatus Class Reference

#include <httpstatus.h>
Detailed Description

HTTP status code enumeration (e.g. 404 Not Found).

(C) 2007 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>status codes</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static Code FromString (const Util::String &amp;str)</code></td>
<td>convert from string</td>
</tr>
<tr>
<td><code>static Util::String ToString (Code c)</code></td>
<td>convert to string</td>
</tr>
<tr>
<td><code>static Util::String ToHumanReadableString (Code c)</code></td>
<td>convert code to human readable string</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpStream
Http::HttpStream Class Reference

#include <httpstream.h>
Detailed Description

Wraps client HTTP requests to a HTTP server into an **IO::Stream**.

(C) 2009 Radon Labs GmbH
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Http::SvgLineChartWriter
Http::SvgLineChartWriter Class Reference

#include <svglinechartwriter.h>

Inheritance diagram for Http::SvgLineChartWriter:
Detailed Description

Specialized SVG page writer which draws a line chart diagrams.

(C) 2008 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SvgLineChartWriter()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~SvgLineChartWriter()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual bool Open()</code></td>
<td>Begin writing to the stream</td>
</tr>
<tr>
<td><code>void SetupXAxis(const Util::String &amp;axisName, const Util::String &amp;unitName, int minVal, int maxVal)</code></td>
<td>Setup the x axis</td>
</tr>
<tr>
<td><code>void SetupYAxis(const Util::String &amp;axisName, const Util::String &amp;unitName, float minVal, float maxVal)</code></td>
<td>Setup the y axis</td>
</tr>
<tr>
<td><code>void AddTrack(const Util::String &amp;name, const Util::String &amp;color, const Util::Array&lt;float&gt;&amp;values)</code></td>
<td>Add a value track</td>
</tr>
<tr>
<td><code>void AddTrack(const Util::String &amp;name, const Util::String &amp;color, const Util::Array&lt;float&gt;&amp;values, const Util::Array&lt;Timing::Tick&gt;&amp;timeStamps)</code></td>
<td>Add a value track with timestamps</td>
</tr>
<tr>
<td><code>void Draw()</code></td>
<td>Draw diagram</td>
</tr>
<tr>
<td><code>void SetCanvasDimensions(SizeT w, SizeT h)</code></td>
<td>Set width and height of canvas in pixels</td>
</tr>
<tr>
<td><code>void SetUnitDimensions(SizeT w, SizeT h)</code></td>
<td>Set width and height in draw units</td>
</tr>
<tr>
<td><code>virtual void Close()</code></td>
<td>End writing to the stream</td>
</tr>
<tr>
<td><code>void WriteContent(const Util::String &amp;text)</code></td>
<td>Write content text</td>
</tr>
</tbody>
</table>
```plaintext
void BeginNode (const Util::String &nodeName)
begin a new node under the current node

void EndNode ()
end current node, set current node to parent

void SetString (const Util::String &name, const Util::String &value)
set string attribute on current node

void BeginTransformGroup (const Math::float2 &translate, float rotate, const Math::float2 &scale)
begin a transformation group

void BeginPaintGroup (const Util::String &fillColor, const Util::String &strokeColor, int strokeWidth)
begin a paint group

void BeginTextGroup (int fontSize, const Util::String &textColor)
begin a text group

void EndGroup ()
end the most recent group

void Rect (float x, float y, float w, float h)
draw a rectangle, all units are in pixels

void Circle (float x, float y, float r)
draw a circle, all units are in pixels

void Ellipse (float x, float y, float rx, float ry)
draw an ellipse, all units are in pixels

void Line (float x1, float y1, float x2, float y2)
draw a line, all units are in pixels

void PolyLine (const Util::Array< Math::float2 > &points)
draw a poly-line, all units are in pixels

void Polygon (const Util::Array< Math::float2 > &points)
draw a polygon, all units are in pixels

void Text (float x, float y, const Util::String &text)
draw text, all units are in pixels

void SetStream (const Ptr< Stream > &s)
set stream to write to
```

const Ptr< Stream > &
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetStream()</code> const</td>
<td>get currently set stream</td>
</tr>
<tr>
<td><code>HasStream()</code> const</td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td><code>IsOpen()</code> const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td><code>GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName()</code> const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC()</code> const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void Http::SvgPageWriter::PolyLine(
    const Util::Array<
        Math::float2
    >& points ) [inherited]
```

draw a poly-line, all units are in pixels

Note: due to a limitation for the length of XML attributes in TinyXML the array size for a single poly primitive should be about 200 max.

```cpp
void Http::SvgPageWriter::Polygon(
    const Util::Array<
        Math::float2
    >& points ) [inherited]
```

draw a polygon, all units are in pixels

Note: due to a limitation for the length of XML attributes in TinyXML the array size for a single poly primitive should be about 200 max.

```cpp
void IO::StreamWriter::SetStream(
    const Ptr<
        Stream
    >& s ) [inherited]
```

set stream to write to

Attaches the writer to a stream. This will increment the refcount of the stream.

Reimplemented in **Messaging::MessageWriter**.

```cpp
const Ptr<
    Stream
>& IO::StreamWriter::GetStream() const [inherited]
```

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use **HasStream()** to determine if a stream is
attached.

bool IO::StreamWriter::HasStream() const [inherited]

return true if a stream is set

Returns true if a stream is attached to the writer.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::SvgPageWriter
Http::SvgPageWriter Class Reference

#include <svgpagewriter.h>

Inheritance diagram for Http::SvgPageWriter:

```
Core::RefCounted
  
IO::StreamWriter
  
Http::SvgPageWriter
  
Http::SvgLineChartWriter
```
Detailed Description

A stream writer to generate simple SVG pages.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SvgPageWriter ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~SvgPageWriter ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>SetCanvasDimensions (SizeT w, SizeT h)</code></td>
<td>set width and height of canvas in pixels</td>
</tr>
<tr>
<td><code>SetUnitDimensions (SizeT w, SizeT h)</code></td>
<td>set width and height in draw units</td>
</tr>
<tr>
<td><code>Open ()</code></td>
<td>begin writing to the stream</td>
</tr>
<tr>
<td><code>Close ()</code></td>
<td>end writing to the stream</td>
</tr>
<tr>
<td><code>WriteContent (const Util::String &amp;text)</code></td>
<td>write content text</td>
</tr>
<tr>
<td><code>BeginNode (const Util::String &amp;nodeName)</code></td>
<td>begin a new node under the current node</td>
</tr>
<tr>
<td><code>EndNode ()</code></td>
<td>end current node, set current node to parent</td>
</tr>
<tr>
<td><code>SetString (const Util::String &amp;name, const Util::String &amp;value)</code></td>
<td>set string attribute on current node</td>
</tr>
<tr>
<td><code>BeginTransformGroup (const Math::float2 &amp;translate, float rotate, const Math::float2 &amp;scale)</code></td>
<td>begin a transformation group</td>
</tr>
<tr>
<td><code>BeginPaintGroup (const Util::String &amp;fillColor, const Util::String &amp;strokeColor, int strokeWidth)</code></td>
<td>begin a paint group</td>
</tr>
<tr>
<td><code>BeginTextGroup (int fontSize, const Util::String &amp;textColor)</code></td>
<td>begin a text group</td>
</tr>
<tr>
<td><code>EndGroup ()</code></td>
<td>end the most recent group</td>
</tr>
<tr>
<td><code>Rect (float x, float y, float w, float h)</code></td>
<td></td>
</tr>
</tbody>
</table>
void Circle (float x, float y, float r)
  draw a circle, all units are in pixels

void Ellipse (float x, float y, float rx, float ry)
  draw an ellipse, all units are in pixels

void Line (float x1, float y1, float x2, float y2)
  draw a line, all units are in pixels

void PolyLine (const Util::Array<Math::float2>&points)
  draw a poly-line, all units are in pixels

void Polygon (const Util::Array<Math::float2>&points)
  draw a polygon, all units are in pixels

void Text (float x, float y, const Util::String&text)
  draw text, all units are in pixels

void SetStream (const Ptr<Stream>&&s)
  set stream to write to

const Ptr<Stream>&& GetStream () const
  get currently set stream

bool HasStream () const
  return true if a stream is set

bool IsOpen () const
  return true if currently open

int GetRefCount () const
  get the current refcount

void AddRef ()
  increment refcount by one

void Release ()
  decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti&rtti) const
  return true if this object is instance of given class

bool IsInstanceOf (const Util::String&className) const
  return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC&classFourCC) const
  return true if this object is instance of given class by fourcc
<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong>(const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong>(const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong>(const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong>( ) const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong>( ) const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

void Http::SvgPageWriter::PolyLine(const Util::Array<Math::float2> & points)
draw a poly-line, all units are in pixels

Note: due to a limitation for the length of XML attributes in TinyXML
the array size for a single poly primitive should be about 200 max.

void Http::SvgPageWriter::Polygon(const Util::Array<Math::float2> & points)
draw a polygon, all units are in pixels

Note: due to a limitation for the length of XML attributes in TinyXML
the array size for a single poly primitive should be about 200 max.

void IO::StreamWriter::SetStream(const Ptr<Stream> & s) [inherited]
set stream to write to

Attaches the writer to a stream. This will increment the refcount of the stream.

Reimplemented in Messaging::MessageWriter.

const Ptr<Stream> & IO::StreamWriter::GetStream() const [inherited]
get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use HasStream() to determine if a stream is
attached.

bool IO::StreamWriter::HasStream() const [inherited]
return true if a stream is set

Returns true if a stream is attached to the writer.

int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks(); [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Input:: GamePad
#include <gamepad.h>

Inheritance diagram for Input::GamePad:
Detailed Description

An input handler which represents a game pad.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>enum</strong> Button</td>
<td><em>gamepad buttons</em></td>
</tr>
<tr>
<td><strong>enum</strong> Axis</td>
<td><em>gamepad axis</em></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><strong>IsConnected</strong> (const)</td>
<td>return true if this game pad is currently connected</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetPlayerIndex</strong> (IndexT i)</td>
<td>set player index -&gt; TODO make threadsafe</td>
</tr>
<tr>
<td>IndexT</td>
<td><strong>GetPlayerIndex</strong> (const)</td>
<td>get the player index of this game pad</td>
</tr>
<tr>
<td>bool</td>
<td><strong>ButtonPressed</strong> (Button btn) const</td>
<td>return true if a button is currently pressed</td>
</tr>
<tr>
<td>bool</td>
<td><strong>ButtonDown</strong> (Button btn) const</td>
<td>return true if button was down at least once in current frame</td>
</tr>
<tr>
<td>bool</td>
<td><strong>ButtonUp</strong> (Button btn) const</td>
<td>return true if button was up at least once in current frame</td>
</tr>
<tr>
<td>float</td>
<td><strong>GetAxisValue</strong> (Axis axis) const</td>
<td>get current axis value</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetLowFrequencyVibrator</strong> (float f)</td>
<td>set low-frequency vibration effect (0.0f .. 1.0f)</td>
</tr>
<tr>
<td>float</td>
<td><strong>GetLowFrequencyVibrator</strong> (const)</td>
<td>get low-frequency vibration</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetHighFrequencyVibrator</strong> (float f)</td>
<td>set high-frequency vibration effect (0.0f .. 1.0f)</td>
</tr>
<tr>
<td>float</td>
<td><strong>GetHighFrequencyVibrator</strong> (const)</td>
<td>get high-frequency vibration</td>
</tr>
<tr>
<td><strong>Util::Array&lt; Input::InputEvent &gt;</strong></td>
<td><strong>GetStateAsInputEvents</strong> (const)</td>
<td>get current state as an array of input events (override in subclass!)</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsAttached</strong> (const)</td>
<td>return true if the input handler is currently attached</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>BeginCapture</strong> ()</td>
<td>capture input to this event handler</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>EndCapture</strong> ()</td>
<td>end input capturing to this event handler</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsCapturing</strong> (const)</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
<td></td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
<td></td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetClassName</strong> () const</td>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td>Util::FourCC <strong>GetClassFourCC</strong> () const</td>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Util::String</td>
<td><strong>ButtonText</strong> (Button btn)</td>
<td>convert button code to string</td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>AxisAsString</strong> (Axis a)</td>
<td>convert axis to string</td>
</tr>
<tr>
<td>static SizeT</td>
<td><strong>GetMaxNumPlayers</strong> ()</td>
<td>get maximum number of players</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void OnAttach ()</td>
<td>called when the handler is attached to the input server</td>
</tr>
<tr>
<td>void OnBeginFrame ()</td>
<td>called on InputServer::BeginFrame()</td>
</tr>
<tr>
<td>void UpdateButtonState (const XINPUT_GAMEPAD &amp;curState, WORD xiBtn, Button btn)</td>
<td>update the state of a game pad button</td>
</tr>
<tr>
<td>void UpdateTriggerAxis (const XINPUT_GAMEPAD &amp;curState, Axis axis)</td>
<td>update the state of a trigger axis</td>
</tr>
<tr>
<td>void UpdateThumbAxis (const XINPUT_GAMEPAD &amp;curState, Axis axis)</td>
<td>update the state of a thumb stick axis</td>
</tr>
<tr>
<td>virtual void OnReset ()</td>
<td>reset the input handler</td>
</tr>
<tr>
<td>void OnRemove ()</td>
<td>called when the handler is removed from the input server</td>
</tr>
<tr>
<td>virtual void OnEndFrame ()</td>
<td>called on InputServer::EndFrame();</td>
</tr>
<tr>
<td>virtual void OnObtainCapture ()</td>
<td>called when input handler obtains capture</td>
</tr>
<tr>
<td>virtual void OnReleaseCapture ()</td>
<td>called when input handler looses capture</td>
</tr>
<tr>
<td>virtual bool OnEvent (const InputEvent &amp;inputEvent)</td>
<td>called when an input event should be processed</td>
</tr>
</tbody>
</table>
Member Function Documentation

void XInput::XInputGamePad::OnBeginFrame() [protected, virtual, inherited]
called on InputServer::BeginFrame()

This compares the current state of the game pad against the previous state and sets the internal state accordingly.

FIXME: Calling XInputGetState() on non-connected controllers is very expensive, thus if XInputGetState return ERROR_DEVICE_NOT_CONNECTED, only call XInputGetState() every 2 seconds to check if a device has actually been connected!!!

Reimplemented from Input::InputHandler.

void XInput::XInputGamePad::UpdateButtonState(const XINPUT_GAMEPAD &curState, WORD xiBtn, Button btn) [protected, inherited]

update the state of a game pad button

Compares the previous and current state of a game pad button and updates the parent class' state accordingly.

Array<InputEvent> Base::GamePadBase::GetStateAsInputEvents() const [inherited]

get current state as an array of input events (override in subclass!)

This method should return the current state of the game pad as input events. It is up to a specific subclass to implement this method.

void Input::InputHandler::BeginCapture() [virtual, inherited]
capture input to this event handler

Begin capturing input to this input handler. This method must be overridden in a subclass, the derived method must call ObtainMouseCapture(), ObtainKeyboardCapture(), or both, depending on what type input events you want to capture. An input handler which captures input gets all input events of the given type exclusively.

Reimplemented in **Base::KeyboardBase**, and **Base::MouseBase**.

```cpp
void Input::InputHandler::EndCapture()
```

end input capturing to this event handler

End capturing input to this input handler. Override this method in a subclass and release the captures obtained in **BeginCapture()**.

Reimplemented in **Base::KeyboardBase**, and **Base::MouseBase**.

```cpp
int Core::RefCounted::GetRefCount() const
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef()
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release()
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
```
Core::RefCounted::GetClassName ( ) const [inline, inherited]

class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Input::InputEvent
#include <inputevent.h>
Detailed Description

The input events of the Input subsystems. Input events are generated by the InputServer and travel through the input handler chain where they are processed.

(C) 2007 Radon Labs GmbH
# Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th><code>Type</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>input event types</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>InputEvent()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>void SetType (Type t)</code></td>
<td>set event type</td>
</tr>
<tr>
<td><code>Type GetType () const</code></td>
<td>get event type</td>
</tr>
<tr>
<td><code>void SetKey (Key::Code key)</code></td>
<td>set key code</td>
</tr>
<tr>
<td><code>Key::Code GetKey () const</code></td>
<td>get key code</td>
</tr>
<tr>
<td><code>void SetChar (Char chr)</code></td>
<td>set character code</td>
</tr>
<tr>
<td><code>Char GetChar () const</code></td>
<td>get character code</td>
</tr>
<tr>
<td><code>void SetMouseButton (MouseButton::Code button)</code></td>
<td>set button code</td>
</tr>
<tr>
<td><code>MouseButton::Code GetMouseButton () const</code></td>
<td>get button code</td>
</tr>
<tr>
<td><code>void SetDeviceIndex (IndexT i)</code></td>
<td>set optional device index (e.g. player index for game pads)</td>
</tr>
<tr>
<td><code>IndexT GetDeviceIndex () const</code></td>
<td>get device index</td>
</tr>
<tr>
<td><code>void SetAbsMousePos (const Math::float2 &amp;p)</code></td>
<td>set absolute pixel mouse position</td>
</tr>
<tr>
<td><code>const Math::float2 &amp; GetAbsMousePos () const</code></td>
<td>get absolute pixel mouse position</td>
</tr>
<tr>
<td><code>void SetNormMousePos (const Math::float2 &amp;p)</code></td>
<td>set normalized mouse position</td>
</tr>
<tr>
<td><code>const Math::float2 &amp; GetNormMousePos () const</code></td>
<td>get normalized mouse position</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static const char * TypeToString (Type t)

convert type to string

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:46 2010
Input::InputHandler
#include <inputhandler.h>

Inheritance diagram for Input::InputHandler:
Detailed Description

**Input** handlers receive and process input events. Handlers are chained together, sorted by priority, and input events travel from one handler to the next. **Input** events may be blocked by an input handler, so that the blocked events are not passed on to the next lower-priority handlers. Subclasses of **InputHandler** present the received input in more specific ways.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InputHandler ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~InputHandler ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>bool IsAttached () const</code></td>
<td>return true if the input handler is currently attached</td>
</tr>
<tr>
<td><code>virtual void BeginCapture ()</code></td>
<td>capture input to this event handler</td>
</tr>
<tr>
<td><code>virtual void EndCapture ()</code></td>
<td>end input capturing to this event handler</td>
</tr>
<tr>
<td><code>bool IsCapturing () const</code></td>
<td>return true if this input handler captures input</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>
const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void</td>
<td><code>OnAttach()</code></td>
<td>called when the handler is attached to the input server</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnRemove()</code></td>
<td>called when the handler is removed from the input server</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnBeginFrame()</code></td>
<td>called on <code>InputServer::BeginFrame()</code></td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnEndFrame()</code></td>
<td>called on <code>InputServer::EndFrame()</code>;</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnObtainCapture()</code></td>
<td>called when input handler obtains capture</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnReleaseCapture()</code></td>
<td>called when input handlerlooses capture</td>
</tr>
<tr>
<td>virtual bool</td>
<td><code>OnEvent()</code></td>
<td>called when an input event should be processed</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnReset()</code></td>
<td>called when the handler should reset itself</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Input::InputHandler::BeginCapture() [virtual]
capture input to this event handler

Begin capturing input to this input handler. This method must be overridden in a subclass, the derived method must call ObtainMouseCapture(), ObtainKeyboardCapture(), or both, depending on what type input events you want to capture. An input handler which captures input gets all input events of the given type exclusively.

Reimplemented in Base::KeyboardBase, and Base::MouseBase.
```

```cpp
void Input::InputHandler::EndCapture() [virtual]
end input capturing to this event handler

End capturing input to this input handler. Override this method in a subclass and release the captures obtained in BeginCapture().

Reimplemented in Base::KeyboardBase, and Base::MouseBase.
```

```cpp
void Input::InputHandler::OnReset() [protected, virtual]
called when the handler should reset itself

OnReset is called when the app loses or gains focus (amongst other occasions). The input handler should reset its internal state to prevent keys from sticking down, etc...

Reimplemented in Base::GamePadBase, Base::KeyboardBase, and Base::MouseBase.
```

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount
Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name
Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code
Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**Input::InputPriority**
Input::InputPriority Class Reference

#include <inputpriority.h>
Detailed Description

**Input** priorities for input handlers.

(C) 2007 Radon Labs GmbH
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Input::InputServer
#include <inputserver.h>

Inheritance diagram for Input::InputServer:

- Core::RefCounted
- Base::InputServerBase
- Win32::Win32InputServer
- Input::InputServer
Detailed Description

The **InputServer** is the central object of the **Input** subsystem. It mainly manages a prioritized list of input handlers which process incoming input events.

(C) 2007 Radon Labs GmbH
### Public Member Functions

**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InputServer ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~InputServer ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>Open ()</code></td>
<td>open the input server</td>
</tr>
<tr>
<td><code>Close ()</code></td>
<td>close the input server</td>
</tr>
<tr>
<td><code>OnFrame ()</code></td>
<td>call after processing window events (reads DInput raw mouse events)</td>
</tr>
<tr>
<td><code>SetMaxNumLocalPlayers (SizeT maxNumLocalPlayers)</code></td>
<td>set the max number of local players for this application (default is 4)</td>
</tr>
<tr>
<td><code>GetMaxNumLocalPlayers ()</code> const</td>
<td>get the max number of local players</td>
</tr>
<tr>
<td><code>IsOpen ()</code></td>
<td>return true if open</td>
</tr>
<tr>
<td><code>SetQuitRequested (bool b)</code></td>
<td>set the quit requested flag</td>
</tr>
<tr>
<td><code>IsQuitRequested ()</code></td>
<td>return true if some subsystem has requested to quit the app (e.g. Alt-F4)</td>
</tr>
<tr>
<td><code>Reset ()</code></td>
<td>reset input state</td>
</tr>
<tr>
<td><code>GetDefaultKeyboard ()</code> const</td>
<td>get the default keyboard input handler</td>
</tr>
<tr>
<td><code>GetDefaultMouse ()</code></td>
<td>get the default mouse input handler</td>
</tr>
<tr>
<td><code>GetDefaultGamePad (IndexT playerIndex)</code></td>
<td>get default gamepad handler (playerIndex is valid up to MaxNumLocalPlayers)</td>
</tr>
</tbody>
</table>
void AttachInputHandler (Input::InputPriority::Code pri, const Ptr<Input::InputHandler> &inputHandler)
attach an input handler

void RemoveInputHandler (const Ptr<Input::InputHandler> &inputHandler)
remove an input handler

virtual void BeginFrame ()
call before processing window events

void EndFrame ()
call at end of frame

void PutEvent (const Input::InputEvent &ie)
put an input event into the handler chain

void ClearMouseCapture ()
clear the current mouse capture (if exists)

void ClearKeyboardCapture ()
clear the current keyboard capture (if exists)

void ClearCapture ()
clear both mouse and keyboard captures

const Ptr<Input::InputHandler> & GetMouseCaptureHandler () const
return the current mouse capture input handler (return invalid ptr if no capture set)

const Ptr<Input::InputHandler> & GetKeyboardCaptureHandler () const
return the current keyboard capture input handler (return invalid ptr if no capture set)

void ObtainMouseCapture (const Ptr<Input::InputHandler> &inputHandler)
only call from InputHandler: capture mouse input to the given input handler

void ReleaseMouseCapture (const Ptr<Input::InputHandler> &inputHandler)
only call from InputHandler: release mouse capture

void ObtainKeyboardCapture (const Ptr<Input::InputHandler> &inputHandler)
only call from InputHandler: capture keyboard input to the given input handler
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ReleaseKeyboardCapture</strong> (const <strong>Ptr</strong><a href="">Input::InputHandler</a> &amp;inputHandler)</td>
<td>only call from InputHandler: release keyboard capture</td>
</tr>
<tr>
<td><strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><strong>GetClassName</strong> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>GetClassFourCC</strong> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool <code>OpenDInputMouse ()</code></td>
<td>setup the DirectInput mouse device for tracking mouse movement</td>
</tr>
<tr>
<td>void <code>CloseDInputMouse ()</code></td>
<td>shutdown the DirectInput mouse device</td>
</tr>
<tr>
<td>void <code>ReadDInputMouse ()</code></td>
<td>get mouse readings</td>
</tr>
<tr>
<td>const <code>Math::float2 &amp; GetMouseMovement () const</code></td>
<td>get the current mouse movement</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool
Win32::Win32InputServer::OpenDInputMouse( ) [protected, inherited]

setup the DirectInput mouse device for tracking mouse movement

This initializes a DirectInput mouse device in order to track raw mouse movement (WM mouse events stop at the screen borders).

void
Win32::Win32InputServer::CloseDInputMouse( ) [protected, inherited]

shutdown the DirectInput mouse device

Close the DirectInput mouse and DirectInput.

void
Win32::Win32InputServer::ReadDInputMouse( ) [protected, inherited]

get mouse readings

Read data from the DirectInput mouse (relative mouse movement since the last frame).

void
Base::InputServerBase::SetMaxNumLocalPlayers( SizeT num ) [inherited]

set the max number of local players for this application (default is 4)

Setup the maximum number of local players for this application. The default number is 1. This defines the number of game pad objects created and queried.

void
Base::InputServerBase::EndFrame( ) [inherited]

call at end of frame

Call this somewhere towards the end of frame, when it is guaranteed...
that noone needs input anymore.

```cpp
void Base::InputServerBase::PutEvent(const Input::InputEvent ie) [inherited]
```

put an input event into the handler chain

NOTE: MouseMove and RawMouseMove events will be distributed to all input handlers regardless of mouse capture state!

```cpp
void Base::InputServerBase::ClearMouseCapture() [inherited]
```

clear the current mouse capture (if exists)

This clears the currently set mouse capture (if exists).

```cpp
void Base::InputServerBase::ClearKeyboardCapture() [inherited]
```

clear the current keyboard capture (if exists)

This clears the currently set keyboard capture (if exists).

```cpp
void Base::InputServerBase::ClearCapture() [inherited]
```

clear both mouse and keyboard captures

This clears the mouse and keyboards captures, if set.

```cpp
void Base::InputServerBase::ObtainMouseCapture(const Ptr<Input::InputHandler> inputHandler) [inherited]
```

only call from InputHandler: capture mouse input to the given input handler

Obtain the mouse capture. All mouse input will go exclusively to the capture input handler until `ReleaseMouseCapture()` is called.

```cpp
const Ptr<
```
only call from InputHandler: release mouse capture

Release the mouse capture. The provided pointer must match the current capture input handler.

only call from InputHandler: capture keyboard input to the given input handler

Obtain the keyboard capture. All keyboard input will go exclusively to the capture input handler until \texttt{ReleaseKeyboardCapture()} is called.

only call from InputHandler: release keyboard capture

Release the mouse capture. The provided pointer must match the current capture input handler.

get the current refcount

Return the current refcount of the object.

increment refcount by one

Increment the refcount of the object.
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Input::Key
#include <key.h>
Detailed Description

Define standard key codes. Note that these are NOT localized character codes, but are similar to Windows virtual keys.

(C) 2006 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
<th>key codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>Group</td>
<td>key group</td>
</tr>
</tbody>
</table>

### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Util::String</code> <strong>ToString</strong> <em>(Code code)</em></td>
<td>convert to string</td>
</tr>
<tr>
<td>static <code>Code</code> <strong>FromString</strong> <em>(const Util::String &amp;str)</em></td>
<td>convert from string</td>
</tr>
<tr>
<td>static <code>Util::Array&lt;Key::Code&gt;</code> <strong>KeyCodesByGroup</strong> <em>(Key::Group group)</em></td>
<td>get key codes by group</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:46 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Input::Keyboard
#include <keyboard.h>

Inheritance diagram for Input::Keyboard:
Detailed Description

An input handler which represents a keyboard for polling.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void BeginCapture ()</td>
<td>capture input to this event handler</td>
</tr>
<tr>
<td>virtual void EndCapture ()</td>
<td>end input capturing to this event handler</td>
</tr>
<tr>
<td>bool KeyPressed (Input::Key::Code keyCode) const</td>
<td>return true if a key is currently pressed</td>
</tr>
<tr>
<td>bool KeyDown (Input::Key::Code keyCode) const</td>
<td>return true if key was down at least once in current frame</td>
</tr>
<tr>
<td>bool KeyUp (Input::Key::Code keyCode) const</td>
<td>return true if key was up at least once in current frame</td>
</tr>
<tr>
<td>const Util::String &amp; GetCharInput () const</td>
<td>get character input in current frame</td>
</tr>
<tr>
<td>bool IsAttached () const</td>
<td>return true if the input handler is currently attached</td>
</tr>
<tr>
<td>bool IsCapturing () const</td>
<td>return true if this input handler captures input</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>
```
| bool | `IsA` (const `Util::FourCC` &rttiFourCC) | return true if this object is instance of given class, or a derived class, by string |
| const `Util::String` & | `GetClassName` () | get the class name |
| `Util::FourCC` | `GetClassFourCC` () | get the class FourCC code |
```
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnAttach</strong> ()</td>
<td>called when the handler is attached to the input server</td>
</tr>
<tr>
<td><strong>OnBeginFrame</strong> ()</td>
<td>called on InputServer::BeginFrame()</td>
</tr>
<tr>
<td><strong>OnEvent</strong> (const Input::InputEvent &amp;inputEvent)</td>
<td>called when an input event should be processed</td>
</tr>
<tr>
<td><strong>OnObtainCapture</strong> ()</td>
<td>called when input handler obtains capture</td>
</tr>
<tr>
<td><strong>OnReleaseCapture</strong> ()</td>
<td>called when input handler looses capture</td>
</tr>
<tr>
<td><strong>OnReset</strong> ()</td>
<td>reset the input handler</td>
</tr>
<tr>
<td><strong>OnRemove</strong> ()</td>
<td>called when the handler is removed from the input server</td>
</tr>
<tr>
<td><strong>OnEndFrame</strong> ()</td>
<td>called on InputServer::EndFrame();</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String& Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Input:: Mouse
Input::Mouse Class Reference

#include <mouse.h>

Inheritance diagram for Input::Mouse:

- Core::RefCounted
- Input::Input-Handler
- Base::MouseBase
- Win32::Win32Mouse
- Input::Mouse
Detailed Description

An input handler which represents a mouse for polling.

(C) 2007 Radon Labs GmbH
Public Member Functions

const Math::float2 & **GetMovement** () const  
*get mouse movement*

virtual void **BeginCapture** ()  
capture input to this event handler

virtual void **EndCapture** ()  
end input capturing to this event handler

bool **ButtonPressed** (Input::MouseButton::Code btn) const  
return true if button is currently pressed

bool **ButtonDown** (Input::MouseButton::Code btn) const  
return true if button was down at least once in current frame

bool **ButtonUp** (Input::MouseButton::Code btn) const  
return true if button was up at least once in current frame

bool **ButtonDoubleClicked** (Input::MouseButton::Code btn) const  
return true if a button has been double clicked

bool **WheelForward** () const  
return true if mouse wheel rotated forward

bool **WheelBackward** () const  
return true if mouse wheel rotated backward

const Math::float2 & **GetPixelPosition** () const  
*get current absolute mouse position (in pixels)*

const Math::float2 & **GetScreenPosition** () const  
*get current screen space mouse position (0.0 .. 1.0)*

bool **IsAttached** () const  
return true if the input handler is currently attached

bool **IsCapturing** () const  
return true if this input handler captures input

int **GetRefCount** () const  
*get the current refcount*

void **AddRef** ()  
increment refcount by one
<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void OnAttach ()</td>
<td>called when the handler is attached to the input server</td>
</tr>
<tr>
<td>virtual void OnBeginFrame ()</td>
<td>called on InputServer::BeginFrame()</td>
</tr>
<tr>
<td>virtual bool OnEvent (const Input::InputEvent &amp;inputEvent)</td>
<td>called when an input event should be processed</td>
</tr>
<tr>
<td>virtual void OnObtainCapture ()</td>
<td>called when input handler obtains capture</td>
</tr>
<tr>
<td>virtual void OnReleaseCapture ()</td>
<td>called when input handler looses capture</td>
</tr>
<tr>
<td>virtual void OnReset ()</td>
<td>reset the input handler</td>
</tr>
<tr>
<td>virtual void OnRemove ()</td>
<td>called when the handler is removed from the input server</td>
</tr>
<tr>
<td>virtual void OnEndFrame ()</td>
<td>called on InputServer::EndFrame();</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String& Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Input::MouseButton
Input::MouseButton Class Reference

#include <mousebutton.h>
Detailed Description

Mouse button codes and conversion from/to string.

(C) 2006 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>code enums</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static Util::String ToString (Code code)</code></td>
<td>convert to string</td>
</tr>
<tr>
<td><code>static Code FromString (const Util::String &amp;str)</code></td>
<td>convert from string</td>
</tr>
</tbody>
</table>
Interface::InterfaceBase
#include <interfacebase.h>

Inheritance diagram for Interface::InterfaceBase:
Detailed Description

**Base** class for interfaces. An interface is the frontend of a fat thread, visible from all threads in the Nebula3 application. Other threads can send messages to the **Interface** singleton which will dispatch the messages to handlers running in the thread context.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InterfaceBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~InterfaceBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>AttachHandler</strong> (const <strong>Ptr</strong> &lt; Messaging::Handler &gt; &amp;h)</td>
<td>Attach a handler to the port (call before open!)</td>
</tr>
<tr>
<td>virtual void <strong>Open ()</strong></td>
<td>Open the async port</td>
</tr>
<tr>
<td>const <strong>Util::StringAtom</strong> &amp; <strong>GetCompanyName</strong> () const</td>
<td>Get the company name</td>
</tr>
<tr>
<td>const <strong>Util::StringAtom</strong> &amp; <strong>GetAppName</strong> () const</td>
<td>Get the application name</td>
</tr>
<tr>
<td>const <strong>Util::StringAtom</strong> &amp; <strong>GetRootDirectory</strong> () const</td>
<td>Get the root directory</td>
</tr>
<tr>
<td>void <strong>SetHandlerThread</strong> (const <strong>Ptr</strong> &lt; HandlerThreadBase &gt; &amp;handlerThread)</td>
<td>Set pointer to handler thread object (must be derived from HandlerThreadBase)</td>
</tr>
<tr>
<td>const <strong>Ptr</strong> &lt; HandlerThreadBase &gt; &amp; <strong>GetHandlerThread</strong> () const</td>
<td>Get pointer to handler thread object</td>
</tr>
<tr>
<td>virtual void <strong>RemoveHandler</strong> (const <strong>Ptr</strong> &lt; Handler &gt; &amp;h)</td>
<td>Dynamically remove a handler from the port</td>
</tr>
<tr>
<td>virtual void <strong>Close ()</strong></td>
<td>Close the async port</td>
</tr>
<tr>
<td><strong>IsOpen ()</strong> const</td>
<td>Return true if port is open</td>
</tr>
<tr>
<td>**template&lt;class MESSAGETYPE&gt; void <strong>Send</strong> (const <strong>Ptr</strong> &lt; MESSAGETYPE &gt; &amp;msg)</td>
<td>Send an asynchronous message to the port</td>
</tr>
<tr>
<td>**template&lt;class MESSAGETYPE&gt; void <strong>SendWait</strong> (const <strong>Ptr</strong> &lt; MESSAGETYPE &gt; &amp;msg)</td>
<td>Send an asynchronous message to the port</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>send a message and wait for completion</td>
<td></td>
</tr>
<tr>
<td>template&lt;class MESSAGETYPE&gt; void Wait (const Ptr&lt; MESSAGETYPE &gt; &amp;msg)</td>
<td>wait for a message to be handled</td>
</tr>
<tr>
<td>template&lt;class MESSAGETYPE&gt; bool Peek (const Ptr&lt; MESSAGETYPE &gt; &amp;msg)</td>
<td>peek a message whether it has been handled</td>
</tr>
<tr>
<td>template&lt;class MESSAGETYPE&gt; void Cancel (const Ptr&lt; MESSAGETYPE &gt; &amp;msg)</td>
<td>cancel a pending message</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td></td>
</tr>
</tbody>
</table>
get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
template<>
void Messaging::AsyncPort::RemoveHandler(const Ptr<Handler> & h) [virtual, inherited]
dynamically remove a handler from the port
Dynamically remove a message handler.
```

```cpp
template<>
void Messaging::AsyncPort::Close() [virtual, inherited]
close the async port
Closes the async port.
Reimplemented in Graphics::GraphicsInterface.
```

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.
```
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Interface::InterfaceHandlerBase
#include <interfacehandlerbase.h>

Inheritance diagram for Interface::InterfaceHandlerBase:
Detailed Description

Base class for message handlers attached to Interface objects.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InterfaceHandlerBase()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>void SetCompanyName(const Util::StringAtom &amp;companyName)</code></td>
<td>set the company name</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetCompanyName()</code></td>
<td>get the company name</td>
</tr>
<tr>
<td><code>void SetAppName(const Util::StringAtom &amp;appName)</code></td>
<td>set the application name</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetAppName()</code></td>
<td>get the application name</td>
</tr>
<tr>
<td><code>virtual void DoWork()</code></td>
<td>optional &quot;per-frame&quot; DoWork method for continuous handlers</td>
</tr>
<tr>
<td><code>virtual void Open()</code></td>
<td>called once on startup</td>
</tr>
<tr>
<td><code>virtual void Close()</code></td>
<td>called once before shutdown</td>
</tr>
<tr>
<td><code>bool IsOpen()</code></td>
<td>return true if open</td>
</tr>
<tr>
<td><code>virtual bool HandleMessage(const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>handle a message, return true if handled</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>IsInstanceOf</code> (const <code>Util::FourCC</code> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA</code> (const <code>Util::String</code> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName</code> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC</code> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```c++
void Messaging::Handler::Open () [virtual, inherited]
```
called once on startup

Open the handler. This method is called once after the handler has been attached to a port and before the first call to `HandleMessage()`.

Reimplemented in `Debug::DebugHandler`, `Http::HttpMessageHandler`, `IO::IOInterfaceHandler`, `Animation::AnimEventServer`, and `Graphics::GraphicsHandler`.

```c++
void Messaging::Handler::Close () [virtual, inherited]
```
called once before shutdown

Close the handler. This method is called once before the handler is detached from the port.

Reimplemented in `Debug::DebugHandler`, `Http::HttpMessageHandler`, `IO::IOInterfaceHandler`, `Animation::AnimEventServer`, and `Graphics::GraphicsHandler`.

```c++
bool Messaging::Handler::HandleMessage (const Ptr< Message > & msg ) [virtual, inherited]
```
handle a message, return true if handled

Derive this method in a subclass to handle specific messages. The method should return true only if a message has been handled.

Reimplemented in `Http::HttpMessageHandler`, `IO::IOInterfaceHandler`, `Animation::AnimEventServer`, `Debug::DebugGraphicsHandler`, and `Graphics::GraphicsHandler`. 
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application
exits.
InternalGraphics::AttachmentServer
InternalGraphics::AttachmentServer Class Reference

#include <attachmentserver.h>

Inheritance diagram for InternalGraphics::AttachmentServer:
Detailed Description

Renderthread side server for managing attachments. Is call once per frame. Manages the updates of the positions of the attached entities, the attaching and detaching of attachments

(C) 2008 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttachmentRotation</td>
<td><em>how to rotate the attached entity</em></td>
</tr>
<tr>
<td>ClearType</td>
<td><em>types for clearing attachments</em></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AttachmentServer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~AttachmentServer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>bool <strong>Open ()</strong></td>
<td>open the server</td>
</tr>
<tr>
<td>void <strong>Close ()</strong></td>
<td>close the server</td>
</tr>
<tr>
<td>bool <strong>isOpen ()</strong> const</td>
<td>return true if open</td>
</tr>
<tr>
<td>void <strong>OnFrame (Timing::Time time)</strong></td>
<td>call this once per frame</td>
</tr>
<tr>
<td>void <strong>RemoveInvalidAttachments ()</strong></td>
<td>remove invalid attachments</td>
</tr>
<tr>
<td>void <strong>AttachEntity (const Math::matrix44 &amp;offset, const ClearType &amp;clearType, const Util::StringAtom &amp;joint, const Ptr&lt; InternalGraphicsEntity &gt; &amp;entityToAttach, const Ptr&lt; InternalModelEntity &gt; &amp;baseEntity, bool keepLocal, AttachmentRotation rotation, bool attachedFromOtherThread)</strong></td>
<td>attach a new entity</td>
</tr>
<tr>
<td>void <strong>AttachEntityTemporary (const Math::matrix44 &amp;offset, const ClearType &amp;clearType, const Util::StringAtom &amp;joint, const Ptr&lt; InternalGraphicsEntity &gt; &amp;entityToAttach, const Ptr&lt; InternalModelEntity &gt; &amp;baseEntity, bool keepLocal, Timing::Time duration, AttachmentRotation rotation, bool attachedFromOtherThread)</strong></td>
<td>attach a new entity, and detach it after given time</td>
</tr>
</tbody>
</table>
| void **SwitchEntity (const Util::StringAtom &oldJoint, const Util::StringAtom &newJoint, const Ptr< InternalModelEntity > &baseEntity, const Ptr< InternalGraphicsEntity > &entityToAttach)** | }
<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>switch a attached entity to another joint</td>
<td>void <strong>DetachEntity</strong> (const <strong>ClearType</strong> &amp;clearType, const <strong>Util::StringAtom</strong> &amp;joint, const <strong>Ptr&lt;</strong> InternalGraphicsEntity <strong>&gt;</strong> &amp;entityToAttach, const <strong>Ptr&lt;</strong> InternalModelEntity <strong>&gt;</strong> &amp;baseEntity)</td>
<td>detach an entity</td>
</tr>
<tr>
<td>detach an entity</td>
<td>void <strong>DetachEntity</strong> (const <strong>Ptr&lt;</strong> InternalGraphics::InternalGraphicsEntity <strong>&gt;</strong> &amp;entity)</td>
<td>remove attachment</td>
</tr>
<tr>
<td>hide or show all attachments on entity</td>
<td>void <strong>SetVisibilityOnAttachments</strong> (const <strong>Ptr&lt;</strong> InternalGraphics::InternalModelEntity <strong>&gt;</strong> &amp;baseEntity, bool visible)</td>
<td>hide or show all attachments on entity</td>
</tr>
<tr>
<td>get the current refcount</td>
<td>int <strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>increment refcount by one</td>
<td>void <strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>decrement refcount and destroy object if refcount is zero</td>
<td>void <strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>return true if this object is instance of given class</td>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>return true if this object is instance of given class by string</td>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>return true if this object is instance of given class by fourcc</td>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class</td>
<td>bool <strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td>bool <strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td>bool <strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>get the class name</td>
<td>const <strong>Util::String</strong> &amp; <strong>GetClassName</strong> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC () const</strong></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void InternalGraphics::AttachmentServer::OnFrame (Timing::Time time)
call this once per frame

FIXME: this method depends on valid character skeletons, thus make sure that it is called after character skeletons have been updated in the character server!

int Core::RefCounted::GetRefCount () const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef () [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release () [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName () const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC () const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
InternalGraphics::InternalCameraEntity
InternalGraphics::InternalCameraEntity
Class Reference

#include <internalcameraentity.h>

Inheritance diagram for InternalGraphics::InternalCameraEntity:
Detailed Description

Represents a camera attached to a graphics stage. Any number of cameras can be attached to a stage.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>LinkType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>visibility link types</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>typedef IndexT</th>
<th>Id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a unique id for graphics entities</td>
</tr>
</tbody>
</table>
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InternalCameraEntity ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~InternalCameraEntity ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>bool IsAttachedToView () const</code></td>
<td>return true if camera is attached to a View</td>
</tr>
<tr>
<td>virtual Math::ClipStatus::Type <code>ComputeClipStatus (const Math::bbox &amp;box)</code></td>
<td>compute clip status against bounding box</td>
</tr>
<tr>
<td><code>void SetCameraSettings (const Shared::CameraSettings &amp;camSettings)</code></td>
<td>set new camera settings</td>
</tr>
<tr>
<td><code>const Shared::CameraSettings &amp; GetCameraSettings () const</code></td>
<td>get camera settings</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetProjTransform () const</code></td>
<td>get projection matrix</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetViewTransform () const</code></td>
<td>get view transform (inverse transform)</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetViewProjTransform () const</code></td>
<td>get view projection matrix</td>
</tr>
<tr>
<td><code>const Math::frustum &amp; GetViewFrustum () const</code></td>
<td>get view frustum</td>
</tr>
<tr>
<td>virtual void <code>HandleMessage (const Ptr&lt; Messaging::Message &gt;)</code></td>
<td>handle message</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>handle(message)</code></td>
<td>Handle a message</td>
</tr>
</tbody>
</table>
| `bool IsActive()`                            | Return true if entity is currently active (is between `OnActivate()`/`OnDeactivate()`)
| `bool IsValid()`                             | Return true if entity is current valid (ready for rendering)               |
| `Id GetId()`                                 | Get the graphics entity's unique id                                         |
| `InternalGraphicsEntityType::Code GetType()`| Get the entity type                                                         |
| `void SetTransform(const Math::matrix44 &m)`| Set the entity's world space transform                                      |
| `const Math::matrix44 & GetTransform()`      | Get the entity's world space transform                                      |
| `void SetVisible(bool b)`                    | Set the entity's visibility                                                |
| `bool IsVisible()`                           | Return true if entity is set to visible                                     |
| `const Ptr< InternalStage > & GetStage()`    | Get the stage this entity is attached to                                   |
| `bool IsAttachedToStage()`                   | Return true if entity is attached to stage                                 |
| `Timing::Time GetEntityTime()`               | Get current entity time                                                    |
| `const Math::bbox & GetLocalBoundingBox()`   | Get the local space bounding box                                           |
| `const Math::bbox & GetGlobalBoundingBox()`  | Get bounding box in global space                                           |
| `ClearLinks(LinkType)`                       |                                                                           |
void linkType)
clear all visibility links

void AddLink (LinkType linkType, const Ptr< InternalGraphicsEntity >&entity)
add visibility link

const Util::Array< Ptr< InternalGraphicsEntity > >& GetLinks (LinkType type) const
get visibility links by type

void MarkRemove ()
mark the entity for removal from the stage at the next possible time

bool IsMarkedForRemove () const
return true if this entity has been marked for removal

void AddDeferredMessage (const Ptr< Messaging::Message >&msg)
add a message for deferred handling once the object becomes valid

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>virtual void</strong> OnDeactivate ()</td>
<td>called before entity is destroyed</td>
</tr>
<tr>
<td>void OnAttachToView (const Ptr&lt; InternalView &gt; &amp;view)</td>
<td>called by View when camera is attached to that view</td>
</tr>
<tr>
<td>void OnRemoveFromView (const Ptr&lt; InternalView &gt; &amp;view)</td>
<td>called by View when camera becomes detached from view</td>
</tr>
<tr>
<td><strong>virtual void</strong> OnTransformChanged ()</td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td>void SetSharedData (const Ptr&lt; FrameSync::FrameSyncSharedData &gt; &amp;data)</td>
<td>set pointer to shared data object</td>
</tr>
<tr>
<td>void SetType (InternalGraphicsEntityType::Code t)</td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td>void SetValid (bool b)</td>
<td>set to valid state (when the entity becomes ready for rendering)</td>
</tr>
<tr>
<td>void UpdateClipStatus (Math::ClipStatus::Type c)</td>
<td>update current clip status</td>
</tr>
<tr>
<td>void UpdateTime (Timing::Time time, Timing::Time timeFactor)</td>
<td>update current time</td>
</tr>
<tr>
<td><strong>virtual void</strong> OnActivate ()</td>
<td>called when entity is created</td>
</tr>
<tr>
<td><strong>virtual void</strong> OnAttachToStage (const Ptr&lt; InternalStage &gt; &amp;stage)</td>
<td>called when attached to Stage</td>
</tr>
<tr>
<td><strong>virtual void</strong> OnRemoveFromStage ()</td>
<td>called when removed from Stage</td>
</tr>
<tr>
<td><strong>virtual void</strong> OnSetupSharedData ()</td>
<td>called to setup the client-portion of the shared data object</td>
</tr>
<tr>
<td><strong>virtual void</strong> OnDiscardSharedData ()</td>
<td>called to discard the client-portion of the shared data object</td>
</tr>
<tr>
<td><strong>virtual void</strong> OnResetSharedData ()</td>
<td>called per frame to reset the shared data object</td>
</tr>
<tr>
<td><strong>virtual void</strong> OnShow ()</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>virtual void <strong>OnHide</strong> ()</td>
<td>called when the entity becomes invisible</td>
</tr>
<tr>
<td>virtual void <strong>OnResolveVisibility</strong> ()</td>
<td>called from Render method in internalview</td>
</tr>
<tr>
<td>virtual void <strong>OnCullBefore</strong> (Timing::Time time, Timing::Time globalTimeFactor, IndexT frameIndex)</td>
<td>called before culling on each(!) graphics entity (visible or not!)</td>
</tr>
<tr>
<td>virtual void <strong>OnNotifyCullingVisible</strong> (const Ptr&lt;InternalGraphicsEntity&gt; &amp;observer, IndexT frameIndex)</td>
<td>called when the entity has been found visible during culling, may be called several times per frame!</td>
</tr>
<tr>
<td>virtual void <strong>OnRenderBefore</strong> (IndexT frameIndex)</td>
<td>called right before rendering</td>
</tr>
<tr>
<td>void <strong>SetLocalBoundingBox</strong> (const Math::bbox &amp;b)</td>
<td>set the local space bounding box</td>
</tr>
<tr>
<td>void <strong>UpdateGlobalBoundingBox</strong> ()</td>
<td>update the global bounding box from the transform and local box</td>
</tr>
<tr>
<td>void <strong>HandleDeferredMessages</strong> ()</td>
<td>handle deferred messages (called by subclasses once resources are loaded)</td>
</tr>
</tbody>
</table>
Member Function Documentation

ClipStatus::Type
InternalGraphics::InternalCameraEntity::ComputeClipStatus

const
(Math::bbox box) [virtual]

compute clip status against bounding box

Computes the clip status of a bounding box in global space against the view volume of this camera entity.

Reimplemented from InternalGraphics::InternalGraphicsEntity.

void
InternalGraphics::InternalCameraEntity::HandleMessage

(const
Ptr<Messaging::Message> msg) [virtual]

handle a message

Handle a message, override this method accordingly in subclasses!

Reimplemented from InternalGraphics::InternalGraphicsEntity.

void
InternalGraphics::InternalCameraEntity::OnTransformChanged

(const
) [protected, virtual]

called when transform matrix changed

We need to keep track of modifications of the transformation matrix.

Reimplemented from InternalGraphics::InternalGraphicsEntity.

void
InternalGraphics::InternalGraphicsEntity::AddDeferredMessage

(const
Ptr<Messaging::Message> msg) [:]

add a message for deferred handling once the object becomes valid

Message handlers may decide to defer message handling until the object has become valid.
void InternalGraphics::InternalGraphicsEntity::OnActivate()
[protected, virtual, inherited]
called when entity is created

Activate the entity. This method is called when the entity is created and attached to the graphics server. During OnActivate() the entity should perform any one-time initializations.

Reimplemented in InternalGraphics::InternalModelEntity.

void InternalGraphics::InternalGraphicsEntity::OnAttachToStage(const Ptr<InternalStage> & s)
[protected, virtual, inherited]
called when attached to Stage

This method is called when the graphics entity is attached to a stage. An entity may only be attached to one stage at any time, but can be attached to different stages during its lifetime. Attaching an entity to a stage may be relatively slow because the entity must be inserted into the cell hierarchy.

void InternalGraphics::InternalGraphicsEntity::OnRemoveFromStage()
[protected, virtual, inherited]
called when removed from Stage

This method is called when the graphics entity is removed from a stage.

void InternalGraphics::InternalGraphicsEntity::OnSetupSharedData()
[protected, virtual, inherited]
called to setup the client-portion of the shared data object

This method is called from OnActivate() to setup the shared data object of the entity. The method must call the ClientSetup() method on the sharedData object with the same template type as the main-thread side entity.

Reimplemented in InternalGraphics::InternalModelEntity.
void InternalGraphics::InternalGraphicsEntity::OnDiscardSharedData() [protected, virtual, inherited]
called to discard the client-portion of the shared data object

Called from OnDeactivate() to discard the shared data object of the entity from the client side.

Reimplemented in InternalGraphics::InternalModelEntity.

void InternalGraphics::InternalGraphicsEntity::OnResetSharedData() [protected, virtual, inherited]
called per frame to reset the shared data object

This method is called once per frame before OnCullBefore() to initialize the shared data object with suitable data (which may be overwritten with uptodate-data later in the frame). This is necessary because the SharedData object is double buffered, and thus if an update if missed for one frame invalid data from the previous frame may "leak" into the next frame.

Reimplemented in InternalGraphics::InternalModelEntity.

void InternalGraphics::InternalGraphicsEntity::OnShow() [protected, virtual, inherited]
called when the entity becomes visible

This method is called from the SetVisible() method when the entity changes from invisible to visible state.

Reimplemented in InternalGraphics::InternalModelEntity.

void InternalGraphics::InternalGraphicsEntity::OnHide() [protected, virtual, inherited]
called when the entity becomes invisible

This method is called from the SetVisible() method when the entity changes from visible to invisible state.
Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnResolveVisibility()
[protected, virtual, inherited]
```
called from Render method in internalview

This method is called whenever the internalview comes to its Render method. Override this method in a subclass to define the visible objects. **Models** must add the model instances here. LightEntities must be added to the LightServer and depending on generating shadows to the ShadowServer.

Reimplemented in `InternalGraphics::InternalModelEntity`, and `Lighting::InternalAbstractLightEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnCullBefore(Timing::Time time_,
Timing::Time timeFactor_,
IndexT frameIndex
[protected, virtual, inherited]
```
called before culling on each(!) graphics entity (visible or not!)

This method is called at the beginning of a frame before culling happens on EVERY entity.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnNotifyCullingVisible
(const Ptr<InternalGraphicsEntity> observer &,
IndexT frameIndex)
```
called when the entity has been found visible during culling, may be called several times per frame!

This method is called during visibility linking when an observed entity
is found to be visible by an observer (a camera or a light entity). NOTE that this method will be called several times per frame, so it may be a good idea to use the graphics server's frame counter to protect expensive code from multiple execution.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnRenderBefore(IndexT frameIndex) [protected, virtual, inherited]
```
called right before rendering

This method is called on the entity from `InternalView::Render()` once per frame for every visible entity.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnRenderDebug() [protected, virtual, inherited]
```
called to render a debug visualization of the entity

This method is called on the graphics entity to render a debug visualization of itself.

Reimplemented in `InternalGraphics::InternalModelEntity`, `Lighting::InternalAbstractLightEntity`, `Lighting::InternalGlobalLightEntity`, `Lighting::InternalPointLightEntity`, and `Lighting::InternalSpotLightEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::HandleDeferredMessages() [protected, inherited]
```
handle deferred messages (called by subclasses once resources are loaded)

This method is called once when an object with deferred validation (e.g. ModelEntities) become valid (usually after their resources have finished loading). Any deferred messages will be processed here.
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

`InternalGraphics::InternalGraphicsEntity`
InternalGraphics::InternalGraphicsEntity
Class Reference

#include <internalgraphicsentity.h>

Inheritance diagram for InternalGraphics::InternalGraphicsEntity:
Detailed Description

A graphics entity represents an atomic graphics object which can be attached to a Stage. **Graphics** entities come in three flavours:

- ModelEntity: a visible model instance
- LightEntity: a light source
- CameraEntity: a camera

**Visibility** queries set graphics entities in relation to each other through bidirectional links. A CameraEntity links to all ModelEntities and LightEntities visible through the camera. Since visibility links are bidirectional, ModelEntities and LightEntities also know through which cameras they are visible. LightEntities have links to all ModelEntities they influence, and ModelEntities know by which lights they are lit.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>LinkType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>visibility link types</td>
</tr>
<tr>
<td>typedef</td>
<td><code>Id</code></td>
</tr>
<tr>
<td>typedef</td>
<td><code>Id</code></td>
</tr>
<tr>
<td></td>
<td>a unique id for graphics entities</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InternalGraphicsEntity()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~InternalGraphicsEntity()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>bool IsActive() const</td>
<td>return true if entity is currently active (is between OnActivate()/OnDeactivate())</td>
</tr>
<tr>
<td>bool IsValid() const</td>
<td>return true if entity is current valid (ready for rendering)</td>
</tr>
<tr>
<td>Id GetId() const</td>
<td>get the graphics entity's unique id</td>
</tr>
<tr>
<td>InternalGraphicsEntityType::Code GetType() const</td>
<td>get the entity type</td>
</tr>
<tr>
<td>void SetTransform(const Math::matrix44 &amp;m)</td>
<td>set the entity's world space transform</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; GetTransform() const</td>
<td>get the entity's world space transform</td>
</tr>
<tr>
<td>void SetVisible(bool b)</td>
<td>set the entity's visibility</td>
</tr>
<tr>
<td>bool IsVisible() const</td>
<td>return true if entity is set to visible</td>
</tr>
<tr>
<td>const Ptr&lt; InternalStage &gt; &amp; GetStage() const</td>
<td>get the stage this entity is attached to</td>
</tr>
<tr>
<td>bool IsAttachedToStage() const</td>
<td>return true if entity is attached to stage</td>
</tr>
<tr>
<td>Timing::Time GetEntityTime() const</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>get current entity time</code></td>
<td></td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetLocalBoundingBox()</code></td>
<td>get the local space bounding box</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetGlobalBoundingBox()</code></td>
<td>get bounding box in global space</td>
</tr>
<tr>
<td><code>void ClearLinks (LinkType linkType)</code></td>
<td>clear all visibility links</td>
</tr>
<tr>
<td><code>void AddLink (LinkType linkType, const Ptr&lt;InternalGraphicsEntity&gt; &amp;entity)</code></td>
<td>add visibility link</td>
</tr>
<tr>
<td><code>const Util::Array&lt;Ptr&lt;InternalGraphicsEntity&gt;&gt; &amp; GetLinks (LinkType type)</code></td>
<td>get visibility links by type</td>
</tr>
<tr>
<td><code>virtual Math::ClipStatus::Type ComputeClipStatus (const Math::bbox &amp;box)</code></td>
<td>compute clip status against bounding box</td>
</tr>
<tr>
<td><code>void MarkRemove()</code></td>
<td>mark the entity for removal from the stage at the next possible time</td>
</tr>
<tr>
<td><code>bool IsMarkedForRemove()</code></td>
<td>return true if this entity has been marked for removal</td>
</tr>
<tr>
<td><code>virtual void HandleMessage (const Ptr&lt;Messaging::Message&gt; &amp;msg)</code></td>
<td>handle a message</td>
</tr>
<tr>
<td><code>void AddDeferredMessage (const Ptr&lt;Messaging::Message&gt; &amp;msg)</code></td>
<td></td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>SetSharedData</code> (const <code>Ptr&lt;FrameSync::FrameSyncSharedData&gt;</code> &amp;data)</td>
<td>set pointer to shared data object</td>
</tr>
<tr>
<td>void</td>
<td><code>SetType</code> (InternalGraphicsEntity::Code t)</td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td>void</td>
<td><code>SetValid</code> (bool b)</td>
<td>set to valid state (when the entity becomes ready for rendering)</td>
</tr>
<tr>
<td>void</td>
<td><code>UpdateClipStatus</code> (Math::ClipStatus::Type c)</td>
<td>update current clip status</td>
</tr>
<tr>
<td>void</td>
<td><code>UpdateTime</code> (Timing::Time time, Timing::Time timeFactor)</td>
<td>update current time</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnActivate</code> ()</td>
<td>called when entity is created</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnDeactivate</code> ()</td>
<td>called before entity is destroyed</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnAttachToStage</code> (const <code>Ptr&lt;InternalStage&gt;</code> &amp;stage)</td>
<td>called when attached to Stage</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnRemoveFromStage</code> ()</td>
<td>called when removed from Stage</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnSetupSharedData</code> ()</td>
<td>called to setup the client-portion of the shared data object</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnDiscardSharedData</code> ()</td>
<td>called to discard the client-portion of the shared data object</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnResetSharedData</code> ()</td>
<td>called per frame to reset the shared data object</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnShow</code> ()</td>
<td>called when the entity becomes visible</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnHide</code> ()</td>
<td>called when the entity becomes invisible</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnResolveVisibility</code> ()</td>
<td>called from Render method in internalview</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnTransformChanged</code> ()</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td><strong>OnCullBefore</strong> (Timing::Time time, Timing::Time globalTimeFactor, IndexT frameIndex)</td>
<td>called before culling on each (!) graphics entity (visible or not!)</td>
<td></td>
</tr>
<tr>
<td><strong>OnNotifyCullingVisible</strong> (const Ptr&lt;InternalGraphicsEntity&gt; &amp;observer, IndexT frameIndex)</td>
<td>called when the entity has been found visible during culling, may be called several times per frame!</td>
<td></td>
</tr>
<tr>
<td><strong>OnRenderBefore</strong> (IndexT frameIndex)</td>
<td>called right before rendering</td>
<td></td>
</tr>
<tr>
<td><strong>OnRenderDebug</strong> ()</td>
<td>called to render a debug visualization of the entity</td>
<td></td>
</tr>
<tr>
<td><strong>SetLocalBoundingBox</strong> (const Math::bbox &amp;b)</td>
<td>set the local space bounding box</td>
<td></td>
</tr>
<tr>
<td><strong>UpdateGlobalBoundingBox</strong> ()</td>
<td>update the global bounding box from the transform and local box</td>
<td></td>
</tr>
<tr>
<td><strong>HandleDeferredMessages</strong> ()</td>
<td>handle deferred messages (called by subclasses once resources are loaded)</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

**ClipStatus::Type**
InternalGraphics::InternalGraphicsEntity::ComputeClipStatus(const Math::bbox box) [virtual]

compute clip status against bounding box

Compute the clip status between this entity and a bounding box in global space. This method must be overwritten in a derived class.

Reimplemented in InternalGraphics::InternalCameraEntity, Lighting::InternalGlobalLightEntity, Lighting::InternalPointLightEntity, and Lighting::InternalSpotLightEntity.

**void**
InternalGraphics::InternalGraphicsEntity::HandleMessage(const Ptr.Messaging::Message msg) [virtual]

handle a message

Handle a message, override this method accordingly in subclasses!


**void**
InternalGraphics::InternalGraphicsEntity::AddDeferredMessage(const Ptr.Messaging::Message msg) [virtual]

add a message for deferred handling once the object becomes valid

Message handlers may decide to defer message handling until the object has become valid.
void InternalGraphics::InternalGraphicsEntity::OnActivate()

[protected, virtual]
called when entity is created

Activate the entity. This method is called when the entity is created and attached to the graphics server. During **OnActivate()** the entity should perform any one-time initializations.

Reimplemented in **InternalGraphics::InternalModelEntity**.

void InternalGraphics::InternalGraphicsEntity::OnDeactivate()

[protected, virtual]
called before entity is destroyed

Deactivate the entity, this method is called when the entity is removed from the graphics server. Any initialization done in **OnActivate()** should be undone here.

Reimplemented in **InternalGraphics::InternalCameraEntity**, and **InternalGraphics::InternalModelEntity**.

void InternalGraphics::InternalGraphicsEntity::OnAttachToStage(const Ptr<InternalStage>& s)

[protected, virtual]
called when attached to Stage

This method is called when the graphics entity is attached to a stage. An entity may only be attached to one stage at any time, but can be attached to different stages during its lifetime. Attaching an entity to a stage may be relatively slow because the entity must be inserted into the cell hierarchy.

void InternalGraphics::InternalGraphicsEntity::OnRemoveFromStage()

[protected, virtual]
called when removed from Stage

This method is called when the graphics entity is removed from a stage.
called to setup the client-portion of the shared data object

This method is called from `OnActivate()` to setup the shared data object of the entity. The method must call the `ClientSetup()` method on the `sharedData` object with the same template type as the main-thread side entity.

Reimplemented in `InternalGraphics::InternalModelEntity`.

called to discard the client-portion of the shared data object

Called from `OnDeactivate()` to discard the shared data object of the entity from the client side.

Reimplemented in `InternalGraphics::InternalModelEntity`.

called per frame to reset the shared data object

This method is called once per frame before `OnCullBefore()` to initialize the shared data object with suitable data (which may be overwritten with up-to-date data later in the frame). This is necessary because the `SharedData` object is double buffered, and thus if an update if missed for one frame invalid data from the previous frame may "leak" into the next frame.

Reimplemented in `InternalGraphics::InternalModelEntity`.

called when the entity becomes visible

This method is called from the `SetVisible()` method when the entity
changes from invisible to visible state.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void
InternalGraphics::InternalGraphicsEntity::OnHide( ) [protected, virtual]
```
called when the entity becomes invisible

This method is called from the `SetVisible()` method when the entity changes from visible to invisible state.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void
InternalGraphics::InternalGraphicsEntity::OnResolveVisibility( ) [protected, virtual]
```
called from Render method in internalview

This method is called whenever the the internalview comes to its Render method. Override this method in a subclass to define the visible objects. **Models** must add the modelinstances here. LightEntities must be added to the LightServer and depending on generating shadows to the ShadowServer.

Reimplemented in `InternalGraphics::InternalModelEntity`, and `Lighting::InternalAbstractLightEntity`.

```cpp
void
InternalGraphics::InternalGraphicsEntity::OnTransformChanged( ) [protected, virtual]
```
called when transform matrix changed

This method is called whenever the transformation matrix has changed by a call to `SetTransform()`. Override this method in a subclass if it wants to be informed about changes to the transformation matrix.

Reimplemented in `InternalGraphics::InternalCameraEntity`, `InternalGraphics::InternalModelEntity`, `Lighting::InternalAbstractLightEntity`, ...
Lighting::InternalGlobalLightEntity, Lighting::InternalPointLightEntity, and Lighting::InternalSpotLightEntity.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnCullBefore (Timing::Time time_,
    Timing::Time timeFactor_,
    IndexT frameIndex)
```

called before culling on each(!) graphics entity (visible or not!)

This method is called at the beginning of a frame before culling happens on EVERY entity.

Reimplemented in InternalGraphics::InternalModelEntity.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnNotifyCullingVisible (const Ptr<InternalGraphicsEntity> observer &,
    IndexT frameIndex)
```

called when the entity has been found visible during culling, may be called several times per frame!

This method is called during visibility linking when an observed entity is found to be visible by an observer (a camera or a light entity). NOTE that this method will be called several times per frame, so it may be a good idea to use the graphics server's frame counter to protect expensive code from multiple execution.

Reimplemented in InternalGraphics::InternalModelEntity.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnRenderBefore (IndexT frameIndex)
```

called right before rendering

This method is called on the entity from InternalView::Render() once per frame for every visible entity.
Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnRenderDebug() [protected, virtual]
called to render a debug visualization of the entity
This method is called on the graphics entity to render a debug visualization of itself.
```

Reimplemented in `InternalGraphics::InternalModelEntity`, `Lighting::InternalAbstractLightEntity`, `Lighting::InternalGlobalLightEntity`, `Lighting::InternalPointLightEntity`, and `Lighting::InternalSpotLightEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::HandleDeferredMessages() [protected]
handle deferred messages (called by subclasses once resources are loaded)
This method is called once when an object with deferred validation (e.g. ModelEntities) become valid (usually after their resources have finished loading). Any deferred messages will be processed here.
```

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName () const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC () const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks () [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**InternalGraphics::InternalGraphicsEntityType**
InternalGraphics::InternalGraphicsEntityType

Class Reference

#include <internalgraphicsentitytype.h>
Detailed Description

Defines graphics entity types.

(C) 2008 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enumeration</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:46 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

InternalGraphics::InternalGraphicsServer
InternalGraphics::InternalGraphicsServer
Class Reference

#include <internalgraphicsserver.h>

Inheritance diagram for InternalGraphics::InternalGraphicsServer:
Detailed Description

The graphics server maintains a the "graphics world" consisting of one or more "stages" and one or more "views" which are attached to the stages.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InternalGraphicsServer()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~InternalGraphicsServer()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>open the graphics server</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>close the graphics server</td>
</tr>
<tr>
<td><code>IsOpen()</code></td>
<td>return true if graphics server is open</td>
</tr>
<tr>
<td><code>CreateStage(const Util::StringAtom &amp;name, const Util::Array&lt;Ptr&lt;Visibility::VisibilitySystemBase&gt; &amp;visSystems)</code></td>
<td>create a stage object</td>
</tr>
<tr>
<td><code>DiscardStage(const Ptr&lt;InternalStage&gt; &amp;stage)</code></td>
<td>discard a stage object</td>
</tr>
<tr>
<td><code>DiscardAllStages()</code></td>
<td>discard all stage objects</td>
</tr>
<tr>
<td><code>HasStage(const Util::StringAtom &amp;name)</code></td>
<td>return true if a stage exists by name</td>
</tr>
<tr>
<td><code>GetStageByName(const Util::StringAtom &amp;name)</code></td>
<td>lookup a stage by name</td>
</tr>
<tr>
<td><code>GetStages()</code></td>
<td>get all stages</td>
</tr>
<tr>
<td><code>CreateView(const Core::Rtti &amp;viewClass, const Util::StringAtom &amp;name, bool isDefaultView=false)</code></td>
<td>create a view object</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void DiscardView (const Ptr&lt;InternalView&gt; &amp;view)</code></td>
<td>discard a view object</td>
</tr>
<tr>
<td><code>void DiscardAllViews ()</code></td>
<td>discard all view objects</td>
</tr>
<tr>
<td><code>bool HasView (const Util::StringAtom &amp;name) const</code></td>
<td>return true if a view exists by name</td>
</tr>
<tr>
<td><code>const Ptr&lt;InternalView&gt; &amp; GetViewByName (const Util::StringAtom &amp;name) const</code></td>
<td>lookup a view by name</td>
</tr>
<tr>
<td><code>const Util::Array&lt;Ptr&lt;InternalView&gt;&gt; &amp; GetViews () const</code></td>
<td>get all views</td>
</tr>
<tr>
<td><code>void SetDefaultView (const Ptr&lt;InternalView&gt; &amp;defView)</code></td>
<td>set the default view</td>
</tr>
<tr>
<td><code>const Ptr&lt;InternalView&gt; &amp; GetDefaultView () const</code></td>
<td>get the default view</td>
</tr>
<tr>
<td><code>bool HasEntity (InternalGraphicsEntity::Id id) const</code></td>
<td>return true if an entity exists by its unique id</td>
</tr>
<tr>
<td><code>const Ptr&lt;InternalGraphicsEntity&gt; &amp; GetEntityByld (InternalGraphicsEntity::Id id) const</code></td>
<td>lookup an entity by its unique id</td>
</tr>
<tr>
<td><code>const Util::Array&lt;Ptr&lt;InternalGraphicsEntity&gt;&gt; &amp; GetEntities () const</code></td>
<td>get all entities</td>
</tr>
<tr>
<td><code>void OnFrame (Timing::Time curTime, Timing::Time globalTimeFactor)</code></td>
<td>call per-frame, this renders the default view</td>
</tr>
<tr>
<td><code>void SetRenderDebug (bool b)</code></td>
<td>set render debug flag</td>
</tr>
<tr>
<td><code>bool GetRenderDebug () const</code></td>
<td>get render debug flag</td>
</tr>
<tr>
<td><code>GetCurrentGlobalLightEntity</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Ptr&lt; Lighting::InternalGlobalLightEntity &gt;</code></td>
<td>const</td>
</tr>
<tr>
<td>get the current global light entity (may return INVALID ptr)</td>
<td></td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

`Ptr< Lighting::InternalGlobalLightEntity >`  
`InternalGraphics::InternalGraphicsServer::GetCurrentGlobalLightEntity ( ) const`

gets the current global light entity (may return INVALID ptr)

FIXME: iterating stuff is terrible!

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```

get the class FourCC code
Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks()
```

[dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)]

This method should be called as the very last before an application exits.
InternalGraphics::InternalModelEntity
InternalGraphics::InternalModelEntity
Class Reference

#include <internalmodelentity.h>

Inheritance diagram for InternalGraphics::InternalModelEntity:
Detailed Description

Represents a visible graphics object.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum LinkType</td>
<td>visibility link types</td>
</tr>
<tr>
<td>typedef IndexT</td>
<td>Id</td>
</tr>
<tr>
<td></td>
<td>a unique id for graphics entities</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InternalModelEntity ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~InternalModelEntity ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>void SetResourceId (const Resources::ResourceId &amp;resId)</strong></td>
<td>Set the model's resource id</td>
</tr>
<tr>
<td><strong>const Resources::ResourceId &amp; GetResourceId () const</strong></td>
<td>Get the model's resource id</td>
</tr>
<tr>
<td><strong>void SetRootNodePath (const Util::StringAtom &amp;rootNodePath)</strong></td>
<td>Optional path to root node (allows to instantiate ModelEntity from part of a ModelNode hierarchy)</td>
</tr>
<tr>
<td><strong>const Util::StringAtom &amp; GetRootNodePath () const</strong></td>
<td>Get root node path, return invalid string atom if root node is invalid</td>
</tr>
<tr>
<td><strong>void SetRootNodeOffsetMatrix (const Math::matrix44 &amp;offsetMatrix)</strong></td>
<td>Set optional root node offset matrix</td>
</tr>
<tr>
<td><strong>const Math::matrix44 &amp; GetRootNodeOffsetMatrix ()</strong></td>
<td>Get optional root node offset matrix</td>
</tr>
<tr>
<td><strong>void ConfigureAnimDrivenMotionTracking (bool enabled, const Util::StringAtom &amp;jointName)</strong></td>
<td>Enable anim driven motion tracking</td>
</tr>
<tr>
<td><strong>void ConfigureAnimEventTracking (bool enabled, bool onlyDominatingClip)</strong></td>
<td>Enable anim event tracking</td>
</tr>
<tr>
<td><strong>void ConfigureCharJointTracking (bool enabled, const Util::Array<a href="">Util::StringAtom</a> &amp;jointNames)</strong></td>
<td>Configure joint tracking</td>
</tr>
<tr>
<td><strong>Resources::Resource::State GetModelResourceState () const</strong></td>
<td>Get the state of the contained managed model resource</td>
</tr>
</tbody>
</table>
const Ptr<Models::ModelInstance> & GetModelInstance () const
get pointer to model instance (only valid if already loaded)

bool HasCharacter () const
return true if this is a character

cost Ptr<Characters::CharacterInstance> & GetCharacterInstance () const
get pointer to character instance

cost Ptr<Characters::Character> & GetCharacter () const
get pointer to character

bool IsAnimDrivenMotionTrackingEnabled () const
return true if anim driven motion tracking is enabled

cost Util::StringAtom & GetAnimDrivenMotionJointName const
get anim driven motion joint name

cost Math::vector & GetAnimDrivenMotionVector const
get the current anim-driven-motion tracking vector

bool IsAnimEventTrackingEnabled () const
return true if anim event tracking is enabled

cost Util::Array<Animation::AnimEventInfo> & GetAnimEvents () const
return the array of anim events of the current frame

bool IsCharJointTrackingEnabled () const
return true if joint tracking is enabled

void AddTrackedCharJoint (const Util::StringAtom & jointName);
dynamically add a tracked joint, note: you cannot remove tracked joints once added!

bool IsCharJointDataValid () const
return true if current char joint data is valid for this frame (may change by visibility)

const Util::Array<Shared::CharJointInfo> & GetCharJointInfos () const
get char joint information

virtual void HandleMessage (const Ptr<Messaging::Message> & msg)
handle a message

bool IsActive () const
return true if entity is currently active (is between OnActivate() and OnDeactivate())
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsValid () const</code></td>
<td>return true if entity is current valid (ready for rendering)</td>
</tr>
<tr>
<td><code>Id GetId () const</code></td>
<td>get the graphics entity's unique id</td>
</tr>
<tr>
<td><code>InternalGraphicsEntity::Code GetType () const</code></td>
<td>get the entity type</td>
</tr>
<tr>
<td><code>void SetTransform (const Math::matrix44 &amp;m)</code></td>
<td>set the entity's world space transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetTransform () const</code></td>
<td>get the entity's world space transform</td>
</tr>
<tr>
<td><code>void SetVisible (bool b)</code></td>
<td>set the entity's visibility</td>
</tr>
<tr>
<td><code>bool IsVisible () const</code></td>
<td>return true if entity is set to visible</td>
</tr>
<tr>
<td><code>const Ptr&lt; InternalStage &gt; &amp; GetStage () const</code></td>
<td>get the stage this entity is attached to</td>
</tr>
<tr>
<td><code>bool IsAttachedToStage () const</code></td>
<td>return true if entity is attached to stage</td>
</tr>
<tr>
<td><code>Timing::Time GetEntityTime () const</code></td>
<td>get current entity time</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetLocalBoundingBox ()</code></td>
<td>get the local space bounding box</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetGlobalBoundingBox ()</code></td>
<td>get bounding box in global space</td>
</tr>
<tr>
<td><code>void ClearLinks (LinkType linkType)</code></td>
<td>clear all visibility links</td>
</tr>
<tr>
<td><code>void AddLink (LinkType linkType, const Ptr&lt; InternalGraphicsEntity &gt; entity)</code></td>
<td>add visibility link</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Ptr&lt; InternalGraphicsEntity &gt; &gt; &amp; GetLinks (LinkType type)</code></td>
<td>get visibility links by type</td>
</tr>
<tr>
<td><code>virtual Math::ClipStatus::Type ComputeClipStatus (const Math::bbox &amp;box)</code></td>
<td>compute clip status against bounding box</td>
</tr>
<tr>
<td><code>void MarkRemove ()</code></td>
<td></td>
</tr>
<tr>
<td><strong>Function</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsMarkedForRemove()</code></td>
<td>const return true if this entity has been marked for removal</td>
</tr>
<tr>
<td><code>void AddDeferredMessage(const Messaging::Message &amp;msg)</code></td>
<td>add a message for deferred handling once the object becomes valid</td>
</tr>
<tr>
<td><code>int GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti)</code></td>
<td>const return true if this object is instance of a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName()</code></td>
<td>const get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC()</code></td>
<td>const get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

- `virtual void OnActivate ()`  
  called when entity is created

- `virtual void OnDeactivate ()`  
  called before entity is destroyed

- `virtual void OnSetupSharedData ()`  
  called to setup the client-portion of the shared data object

- `virtual void OnDiscardSharedData ()`  
  called to discard the client-portion of the shared data object

- `virtual void OnResetSharedData ()`  
  called per frame to reset the shared data object

- `virtual void OnResolveVisibility ()`  
  called from internalview

- `virtual void OnTransformChanged ()`  
  called when transform matrix changed

- `virtual void OnCullBefore (Timing::Time time, Timing::Time globalTimeFactor, IndexT frameIndex)`  
  called before culling on each(!) graphics entity (visible or not!)

- `virtual void OnNotifyCullingVisible (const Ptr<InternalGraphicsEntity> &observer, IndexT frameIndex)`  
  called when the entity has been found visible, may be called several times per frame!

- `virtual void OnRenderBefore (IndexT frameIndex)`  
  called right before rendering

- `virtual void OnRenderDebug ()`  
  called to render a debug visualization of the entity

- `void ValidateModelInstance ()`  
  validate the ModelInstance

- `virtual void OnShow ()`  
  called when the entity becomes visible

- `virtual void OnHide ()`  
  called when the entity becomes invisible

- `void HandleCharacterAnimEvents (Timing::Time time)`  
  handle per-frame anim events
void HandleCharacterAnimDrivenMotion ()
handle per-frame anim-driven-movement stuff

void HandleTrackedJoints ()
handle tracked character joints

void SetSharedData (const Ptr<
FrameSync::FrameSyncSharedData > &data)
set pointer to shared data object

void SetType (InternalGraphicsEntityType::Code t)
set entity type, call in constructor of derived class!

void SetValid (bool b)
set to valid state (when the entity becomes ready for rendering)

void UpdateClipStatus (Math::ClipStatus::Type c)
update current clip status

void UpdateTime (Timing::Time time, Timing::Time timeFactor)
update current time

virtual void OnAttachToStage (const Ptr< InternalStage > &stage)
called when attached to Stage

virtual void OnRemoveFromStage ()
called when removed from Stage

void SetLocalBoundingBox (const Math::bbox &b)
set the local space bounding box

void UpdateGlobalBoundingBox ()
update the global bounding box from the transform and local box

void HandleDeferredMessages ()
handle deferred messages (called by subclasses once resources are loaded)
Member Function Documentation

```cpp
void InternalGraphics::InternalModelEntity::HandleMessage(const Ptr<Messaging::Message> &msg) [virtual]
```

handle a message

Handle a message, override this method accordingly in subclasses!

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void InternalGraphics::InternalModelEntity::OnActivate() [protected, virtual]
```

called when entity is created

This will initialize the managed model inside the ModelEntity.

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void InternalGraphics::InternalModelEntity::OnDeactivate() [protected, virtual]
```

called before entity is destroyed

Cleanup our managed model, and ModellInstance if it is already initialized.

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void InternalGraphics::InternalModelEntity::OnSetupSharedData() [protected, virtual]
```

called to setup the client-portion of the shared data object

This method is called from `OnActivate()` to setup the shared data object of the entity. The method must call the ClientSetup() method on the sharedData object with the same template type as the main-thread side entity.
Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void InternalGraphics::InternalModelEntity::OnDiscardSharedData() [protected, virtual]
```
called to discard the client-portion of the shared data object

Called from `OnDeactivate()` to discard the shared data object of the entity from the client side.

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void InternalGraphics::InternalModelEntity::OnResetSharedData() [protected, virtual]
```
called per frame to reset the shared data object

This method is called once per frame before `OnCullBefore()` to initialize the shared data object with suitable data (which may be overwritten with up-to-date-data later in the frame). This is necessary because the SharedData object is double buffered, and thus if an update is missed for one frame invalid data from the previous frame may "leak" into the next frame.

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void InternalGraphics::InternalModelEntity::OnResolveVisibility() [protected, virtual]
```
called from `internalview`

This method is called whenever the the internalview comes to its `Render` method. Add light entities to the LightServer or to the ShadowServer.

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void InternalGraphics::InternalModelEntity::OnTransformChanged() [protected, virtual]
```
called when transform matrix changed
In `OnTransformChanged()` we need to update the transformation of our ModelInstance (if it has already been initialised).

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void InternalGraphics::InternalModelEntity::OnCullBefore(Timing::Time time_,
                                                      Timing::Time timeFactor_,
                                                      IndexT frameIndex) { }
```

called before culling on each(!) graphics entity (visible or not!)

This method is called each frame on each entity (visible or not), so keep this method cheap!

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void InternalGraphics::InternalModelEntity::OnNotifyCullingVisible(const Ptr<InternalGraphicsEntity> observer, 
                                                                   IndexT frameIndex) { }
```

called when the entity has been found visible, may be called several times per frame!

This method is called once (per-view) before the entity is rendered, and only for visible entities. Please note that this method may be called several times per frame!

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void InternalGraphics::InternalModelEntity::OnRenderBefore(IndexT frameIndex) { }
```

called right before rendering

This method is called on the entity from `InternalView::Render()` once per frame for every visible entity.
Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void
InternalGraphics::InternalModelEntity::OnRenderDebug() [protected, virtual]
```
called to render a debug visualization of the entity

Render our debug visualization (the bounding box).

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void
InternalGraphics::InternalModelEntity::ValidateModelInstance() [protected]
```
validate the ModelInstance

This creates and initializes our ModelInstance if needed. Since Model loading happens asynchronously this may happen at any time after the ModelEntity is activated.

```cpp
void
InternalGraphics::InternalModelEntity::OnShow() [protected, virtual]
```
called when the entity becomes visible

This method is called from the `SetVisible()` method when the entity changes from invisible to visible state.

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void
InternalGraphics::InternalModelEntity::OnHide() [protected, virtual]
```
called when the entity becomes invisible

This method is called from the `SetVisible()` method when the entity changes from visible to invisible state.

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void
InternalGraphics::InternalModelEntity::HandleCharacterAnimEvents(Timing::Time time) [prot
handle per-frame anim events

Handle character anim event stuff, always call this method, even if anim event tracking is not enabled.

```cpp
void InternalGraphics::InternalModelEntity::HandleCharacterAnimDrivenMotion() [protected]
```

handle per-frame anim-driven-movement stuff

Handle character anim driven motion feedback. This is called once per frame for visible characters.

```cpp
void InternalGraphics::InternalModelEntity::HandleTrackedJoints() [protected]
```

handle tracked character joints

Handle the tracked character joints if joint tracking is enabled. NOTE: it's important that this method is called after the character's asynchronous update jobs have finished!

```cpp
ClipStatus::Type const InternalGraphics::InternalGraphicsEntity::ComputeClipStatus(const Math::bbox & box) [virtual, inherited]
```

compute clip status against bounding box

Compute the clip status between this entity and a bounding box in global space. This method must be overwritten in a derived class.

Reimplemented in InternalGraphics::InternalCameraEntity, Lighting::InternalGlobalLightEntity, Lighting::InternalPointLightEntity, and Lighting::InternalSpotLightEntity.

```cpp
void InternalGraphics::InternalGraphicsEntity::AddDeferredMessage(const Messaging::Message msg) [inherited]
```

add a message for deferred handling once the object becomes valid
Message handlers may decide to defer message handling until the object has become valid.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnAttachToStage (const Ptr<InternalStage>& s) [protected, virtual, inherited]
```
called when attached to Stage

This method is called when the graphics entity is attached to a stage. An entity may only be attached to one stage at any time, but can be attached to different stages during its lifetime. Attaching an entity to a stage may be relatively slow because the entity must be inserted into the cell hierarchy.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnRemoveFromStage () [protected, virtual, inherited]
```
called when removed from Stage

This method is called when the graphics entity is removed from a stage.

```cpp
void InternalGraphics::InternalGraphicsEntity::HandleDeferredMessages () [protected, inherited]
```
handle deferred messages (called by subclasses once resources are loaded)

This method is called once when an object with deferred validation (e.g. ModelEntities) become valid (usually after their resources have finished loading). Any deferred messages will be processed here.

```cpp
int Core::RefCounted::GetRefCount () const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef () [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
InternalGraphics::InternalStage
InternalGraphics::InternalStage Class Reference

#include <internalstage.h>

Inheritance diagram for InternalGraphics::InternalStage:
Detailed Description

A graphics stage groups graphics entities (models, cameras and lights) together for rendering. The main job of a Stage is to speed up visibility queries between the attached graphics entities. Different visibility query strategies are implemented by Stage subclasses. Nebula3 comes with a set of generic Stage subclasses for different purposes, but applications are free to derive their own subclasses which implement visibility query mechanisms tailored to the application.

**Visibility** queries exist in the following variations:

- Camera->Light: this finds all light entities visible from a given camera
- Camera->Model: this finds all model entities visible from a given camera
- Light->Model: this finds all model entities which are lit by a given light source

Those visibility queries establish so-called visibility links between graphics entities which are then used by the lower level rendering subsystems to speed up rendering.

To render the content of a stage, at least one View object is needed. A View object binds renders a stage through a camera entity into a render target. Any number of View objects can exist in parallel, and may be bound to any Stage. Furthermore, dependencies between View objects may be defined (so that a View object will first ask the View objects it depends on to render themselves).

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InternalStage()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~InternalStage()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>bool <code>IsAttachedToServer()</code> const</td>
<td>return true if currently attached to graphics server</td>
</tr>
<tr>
<td>const <code>Util::StringAtom &amp; GetName()</code> const</td>
<td>get human readable name of the stage</td>
</tr>
<tr>
<td>virtual void <code>AttachEntity(const </code>Ptr<code>&lt; InternalGraphicsEntity &gt; &amp;graphicsEntity)</code></td>
<td>attach an entity to the stage</td>
</tr>
<tr>
<td>virtual void <code>RemoveEntity(const </code>Ptr<code>&lt; InternalGraphicsEntity &gt; &amp;entity)</code></td>
<td>remove an entity from the stage</td>
</tr>
<tr>
<td>virtual void <code>NotifyOfEntityTransformChange(const </code>Ptr<code>&lt; InternalGraphicsEntity &gt; &amp;entity)</code></td>
<td>notify of a transform change</td>
</tr>
<tr>
<td>const <code>Util::Array&lt; </code>Ptr<code>&lt; InternalGraphicsEntity &gt; &gt; &amp; GetEntities()</code> const</td>
<td>get an array of all entities attached to the stage</td>
</tr>
<tr>
<td>const <code>Util::Array&lt; </code>Ptr<code>&lt; InternalGraphicsEntity &gt; &gt; &amp; GetEntitiesByType(InternalGraphicsEntityTypeType type)</code> const</td>
<td>get entities by type</td>
</tr>
<tr>
<td>void <code>RemoveAllEntities()</code></td>
<td>remove all entities</td>
</tr>
<tr>
<td>virtual void <code>OnCullBefore(Timing::Time curTime, Timing::Time globalTimeFactor, IndexT frameIndex)</code></td>
<td>call OnCullBefore in entities in the stage</td>
</tr>
<tr>
<td><code>UpdateCameraLinks(const </code>Ptr<code>&lt; InternalEntity &amp; cameraEntity)</code></td>
<td>update camera links</td>
</tr>
</tbody>
</table>
virtual void InternalCameraEntity &cameraEntity) 
update camera links for a given camera

virtual void UpdateLightLinks ()
update light links for all visible lights (after updating camera links!)

void AttachVisibilitySystems (const Util::Array< Ptr<Visibility::VisibilitySystemBase> &systems) 
attach visibility systems to checker

Visibility::VisibilityChecker & GetVisibilityChecker () 
get visibility checker

void OnRenderDebug ()
on render debug

int GetRefCount () const 
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) 
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const 
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const 
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const 
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const 
return true if this object is instance of given class, or a derived class, by string
<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>SetName</strong> (const <em>Util::StringAtom</em> &amp;name)</td>
<td>set a human readable name on the stage</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnAttachToServer</strong> ()</td>
<td>called when the stage is attached to graphics server</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnRemoveFromServer</strong> ()</td>
<td>called when the stage is detached from graphics server</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void InternalGraphics::InternalStage::AttachEntity(const Ptr<InternalGraphicsEntity> entity) [virtual]
```

attach an entity to the stage

Add an entity to the stage. The method OnAttachToStage() will be invoked on the entity, and the entity will be inserted into the cell hierarchy of the stage (which in turn call OnAttachToCell() on the entity).

```cpp
void InternalGraphics::InternalStage::RemoveEntity(const Ptr<InternalGraphicsEntity> entity) [virtual]
```

remove an entity from the stage

Remove an entity from the stage. This will remove the entity from the cell hierarchy of the stage (which invoked OnRemoveFromCell() on the entity), and then the method OnRemoveFromStage() will be called on the entity.

```cpp
void InternalGraphics::InternalStage::OnCullBefore(Timing::Time curTime,
                                                  Timing::Time globalTimeFactor,
                                                  IndexT frameIndex)
                                                  [virtual]
```

call OnCullBefore in entities in the stage

Call the **OnCullBefore()** method on ALL methods in the stage (no matter whether they are visible or not).

```cpp
void InternalGraphics::InternalStage::UpdateCameraLinks(const Ptr<InternalCameraEntity> cameraEntity) [virtual]
```

update camera links for a given camera
Update visibility links for a given camera. This will create bidirectional visibility links between the camera and all other entities (most importantly light and model entities) which are visible through this camera. This method must be called once for each active camera after UpdateEntities() and before UpdateVisibleLightLinks().

```cpp
void InternalGraphics::InternalStage::UpdateLightLinks() [virtual]
```

Update light links for all visible lights (after updating camera links!)

For each visible light entity, this method will create light links between the light entities, and model entities influenced by this light. This method must be called after `UpdateCameraLinks()` (this makes sure that no invisible lights and models will be checked).

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name
Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

InternalGraphics::InternalView
InternalGraphics::InternalView Class Reference

#include <internalview.h>

Inheritance diagram for InternalGraphics::InternalView:

```
+----------------+      +----------------+
| Core::RefCounted|      | InternalGraphics::InternalView |
```

Detailed Description

A graphics View is used to render a Stage through a CameraEntity into a RenderTarget. Any number of views can be associated with the same Stage. Views may depend on other views. When a View is rendered, it will first ask the Views it depends on to render themselves. Subclasses of View may implement their own rendering strategies.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InternalView ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~InternalView ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>bool IsAttachedToServer () const</code></td>
<td>return true if currently attached to graphics server</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetName () const</code></td>
<td>get human-readable name</td>
</tr>
<tr>
<td><code>void SetStage (const internal::Ptr&lt;internal::Stage&gt; &amp;stage)</code></td>
<td>set the stage this View is associated with</td>
</tr>
<tr>
<td><code>const internal::Ptr&lt;internal::Stage&gt; &amp; GetStage () const</code></td>
<td>get the stage this View is associated with</td>
</tr>
<tr>
<td><code>void SetCameraEntity (const internal::Ptr&lt;internal::CameraEntity&gt; &amp;camera)</code></td>
<td>set the CameraEntity this View looks through</td>
</tr>
<tr>
<td><code>const internal::Ptr&lt;internal::CameraEntity&gt; &amp; GetCameraEntity () const</code></td>
<td>get the CameraEntity this View looks through</td>
</tr>
<tr>
<td><code>void SetRenderTarget (const CoreGraphics::RenderTarget &amp;renderTarget)</code></td>
<td>set the render target this view renders to</td>
</tr>
<tr>
<td><code>const CoreGraphics::RenderTarget &amp; GetRenderTarget () const</code></td>
<td>get the render target this view renders to</td>
</tr>
<tr>
<td><code>bool HasRenderTarget () const</code></td>
<td>check if render target is set</td>
</tr>
<tr>
<td><code>void SetMultipleRenderTarget (const CoreGraphics::MultipleRenderTarget &amp;renderTarget)</code></td>
<td>set the render target this view renders to</td>
</tr>
<tr>
<td><code>const CoreGraphics::MultipleRenderTarget &amp; GetMultipleRenderTarget () const</code></td>
<td>get the render target this view renders to</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Frame::FrameShader</code></td>
<td>set the view's frame shader</td>
</tr>
<tr>
<td><code>GetFrameShader()</code></td>
<td>get the view's frame shader</td>
</tr>
<tr>
<td><code>AddDependency (const Ptr&lt;InternalView&gt; &amp;view)</code></td>
<td>add a view which this view depends on</td>
</tr>
<tr>
<td><code>GetDependencies()</code></td>
<td>get all dependency views</td>
</tr>
<tr>
<td><code>UpdateVisibilityLinks()</code></td>
<td>update the visibility links for this view</td>
</tr>
<tr>
<td><code>ApplyCameraSettings()</code></td>
<td>apply camera settings</td>
</tr>
<tr>
<td><code>Render (IndexT frameIndex)</code></td>
<td>render the view into its render target</td>
</tr>
<tr>
<td><code>RenderDebug()</code></td>
<td>render a debug view of the world</td>
</tr>
<tr>
<td><code>GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class string</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given fourcc</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given derived class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>SetName</strong> (const Util::StringAtom &amp;name)</td>
<td>set a human-readable name of the view</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnAttachToServer</strong> ()</td>
<td>called when attached to graphics server</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnRemoveFromServer</strong> ()</td>
<td>called when detached from graphics server</td>
</tr>
<tr>
<td>void</td>
<td><strong>ResolveVisibleLights</strong> ()</td>
<td>resolve visible lights</td>
</tr>
<tr>
<td>void</td>
<td><strong>ResolveVisibleModelNodeInstances</strong> (IndexT frameIndex)</td>
<td>resolve visibility for optimal batch rendering</td>
</tr>
</tbody>
</table>
Member Function Documentation

void InternalGraphics::InternalView::UpdateVisibilityLinks() [virtual]

update the visibility links for this view

This method updates the visibility links for the view's camera. This method should usually be called before \texttt{Render()}.

void InternalGraphics::InternalView::ApplyCameraSettings()

apply camera settings

This method sets the new camera transforms valid for this frame in the transform device

void InternalGraphics::InternalView::Render(IndexT frameIndex) [virtual]

render the view into its render target

This method renders the current view into the render target. This method must be called sometimes after \texttt{UpdateVisibilityLinks()}

void InternalGraphics::InternalView::RenderDebug() [virtual]

render a debug view of the world

Renders a debug visualization. Can be called alone or after \texttt{Render()}.

void InternalGraphics::InternalView::ResolveVisibleLights()

resolve visible lights

This attaches visible lights to the light and shadow server.
Todo:
: currently this methods needs to go over all visible graphics entities to find the lights...

```cpp
void InternalGraphics::InternalView::ResolveVisibleModelNodeInstances(IndexT frameIndex) [protected]

resolve visibility for optimal batch rendering

Resolve all visible ModelNodeInstances by following the visibility links of our camera. This is necessary as preparation for rendering. This is also where the OnRenderBefore() callback will be invoked on entities.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC () const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

InternalGraphics::VisibilityCell
InternalGraphics::VisibilityCell Class Reference

#include <visibilitycell.h>
Detailed Description

Hierarchies of **VisibilityCell** objects group graphics entities by spatial relationship. They are the key class for efficient visibility queries. An application may derive specialized subclasses of **VisibilityCell** which must only adhere to the following 2 simple rules:

- if the **VisibilityCell** object is fully visible, all child VisibilityCells and all Entities attached to the **VisibilityCell** are guaranteed to be visible
- if the **VisibilityCell** object is fully invisible, all child VisibilityCells and all Entities attached to the **VisibilityCell** are guaranteed to be invisible

**Todo:**
- need to handle extra shadow bounding box
- statistics and profiling
- need to add visibility depending on LOD
- multithreaded visibility link update?
- add dirty handling to visibility links (e.g. don't need to update links between static lights and static objects)

(C) 2010 Radon Labs GmbH
IO::Archive
#include <archive.h>

Inheritance diagram for IO::Archive:
Detailed Description

Wrapper class for a platform-specific file archive.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>Setup (const URI &amp;uri)</td>
<td>setup the archive from an URI</td>
</tr>
<tr>
<td>void</td>
<td>Discard ()</td>
<td>discard the archive</td>
</tr>
<tr>
<td>Util::Array&lt; Util::String &gt;</td>
<td>ListFiles (const Util::String &amp;dirPathInArchive, const Util::String &amp;pattern) const</td>
<td>list all files in a directory in the archive</td>
</tr>
<tr>
<td>Util::Array&lt; Util::String &gt;</td>
<td>ListDirectories (const Util::String &amp;dirPathInArchive, const Util::String &amp;pattern) const</td>
<td>list all subdirectories in a directory in the archive</td>
</tr>
<tr>
<td>URI</td>
<td>ConvertToArchiveURI (const URI &amp;fileURI)</td>
<td>convert a “file:” URI into a &quot;zip:&quot; URI pointing into this archive</td>
</tr>
<tr>
<td>Util::String</td>
<td>ConvertToPathInArchive (const Util::String &amp;absPath) const</td>
<td>convert an absolute path to local path inside archive, returns empty string if absPath doesn't point into this archive</td>
</tr>
<tr>
<td>bool</td>
<td>IsValid ()</td>
<td>return true if archive is valid</td>
</tr>
<tr>
<td>const URI &amp;</td>
<td>GetURI ()</td>
<td>get the URI of the archive</td>
</tr>
<tr>
<td>int</td>
<td>GetRefCount ()</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td>AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void</td>
<td>Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;className)</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC</td>
<td>return true if this object is instance of given class by FourCC</td>
</tr>
<tr>
<td>Function</td>
<td>Signature</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>bool &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
<td></td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
<td></td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks ()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool IO::ZipArchive::Setup(const URI & zipFileURI) [inherited]
```

setup the archive from an URI

This opens the zip archive and reads the table of content as a tree of ZipDirEntry and ZipFileEntry objects.

Reimplemented from IO::ArchiveBase.

```cpp
void IO::ZipArchive::Discard() [inherited]
```

discard the archive

This closes the zip archive, releasing the table of contents and closing the zip file.

Reimplemented from IO::ArchiveBase.

```cpp
URI IO::ZipArchive::ConvertToArchiveURI(const URI & fileURI) const [inherited]
```

convert a "file:" URI into a "zip:" URI pointing into this archive

This method takes a normal "file:" scheme URI and convertes it into a "zip:" scheme URI which points to the file in this zip archive. This is used by the IoServer for transparent file access into zip archives.

Reimplemented from IO::ArchiveBase.

```cpp
String IO::ZipArchive::ConvertToPathInArchive(const Util::String absPath) const [inherited]
```

convert an absolute path to local path inside archive, returns empty string if absPath doesn't point into this archive
Test if an absolute path points into the zip archive and return a locale path into the zip archive. This will not test, whether the file or directory inside the zip archive actually exists, only if the path points INTO the zip archive by checking against the location directory of the zip archive.

Reimplemented from IO::ArchiveBase.

int Core::RefCounted::GetRefCount(  ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef(  ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release(  ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName(  ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC(  ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
IO::ArchiveBase

Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields
IO::ArchiveBase Class Reference

#include <archivebase.h>

Inheritance diagram for IO::ArchiveBase:

```
Core::RefCounted
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
IO::ArchiveBase
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
IO::ZipArchive
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
IO::Archive
```
Detailed Description

**Base** class of file archives. Subclasses of this class implemented support for specific archive formats, like zip.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ArchiveBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~ArchiveBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>bool Setup (const URI &amp;archiveURI)</strong></td>
<td>Setup the archive from an <strong>URI</strong> (without file extension)</td>
</tr>
<tr>
<td><strong>void Discard ()</strong></td>
<td>Discard the archive</td>
</tr>
<tr>
<td><strong>bool IsValid () const</strong></td>
<td>Return true if archive is valid</td>
</tr>
<tr>
<td><strong>const URI &amp; GetURI () const</strong></td>
<td>Get the <strong>URI</strong> of the archive</td>
</tr>
<tr>
<td><strong>Util::Array<a href="">Util::String</a></strong> ListFiles()</td>
<td>List all files in a directory in the archive</td>
</tr>
<tr>
<td><strong>Util::Array<a href="">Util::String</a></strong> ListDirectories()</td>
<td>List all subdirectories in a directory in the archive</td>
</tr>
<tr>
<td><strong>const URI</strong> ConvertToArchiveURI(const URI &amp;fileURI)**</td>
<td>Convert a “file:” <strong>URI</strong> into a archive-specific <strong>URI</strong> pointing into this archive</td>
</tr>
<tr>
<td><strong>const Util::String</strong> ConvertToPathInArchive(const Util::String &amp;absPath)**</td>
<td>Convert an absolute path to local path inside archive, returns empty string if absPath doesn't point into this archive</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti)</strong></td>
<td>Is this instance of the given <strong>Rtti</strong>?</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Util::String</code> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Util::FourCC</code> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Util::String</code> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp; <code>GetClassName</code> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC</code> <code>GetClassFourCC</code> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool IO::ArchiveBase::Setup(const URI & archiveFileURI)
```

setup the archive from an **URI** (without file extension)

Setup the archive object from an **URI** pointing to an archive file. This method may return false if something went wrong (archive file not found, or wrong format).

Reimplemented in **IO::ZipArchive**.

```cpp
Array<String> IO::ArchiveBase::ListFiles(const Util::String & dirPathInArchive, const Util::String & pattern)
```

list all files in a directory in the archive

List all files in an archive directory. Override this method in a subclass!

Reimplemented in **IO::ZipArchive**.

```cpp
Array<String> IO::ArchiveBase::ListDirectories(const Util::String & dirPathInArchive, const Util::String & pattern)
```

list all subdirectories in a directory in the archive

List all directories in an archive directory. Override this method in a subclass!

Reimplemented in **IO::ZipArchive**.
convert a "file:" URI into a archive-specific URI pointing into this archive

This method should convert a "file:" URI into an URI suitable for an archive specific stream class.

Override this method in a subclass!

Reimplemented in IO::ZipArchive.

convert an absolute path to local path inside archive, returns empty string if absPath doesn't point into this archive

This method should convert an absolute file system path into a local path in the archive suitable for ListFiles() and ListDirectories().

Override this method in a subclass!

Reimplemented in IO::ZipArchive.

get the current refcount

Return the current refcount of the object.

increment refcount by one

Increment the refcount of the object.
void Core::RefCounted::Release

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::ArchiveFileSystem
# include <archivefilesystem.h>

Inheritance diagram for IO::ArchiveFileSystem:
Detailed Description

Top-level platform wrapper class of archive file systems.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ArchiveFileSystem ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~ArchiveFileSystem ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <strong>Setup ()</strong></td>
<td>setup the archive file system</td>
</tr>
<tr>
<td>void <strong>Discard ()</strong></td>
<td>discard the archive file system</td>
</tr>
<tr>
<td><strong>Ptr&lt; Archive &gt;</strong> <strong>FindArchiveWithFile</strong> (const <strong>URI</strong> &amp;fileUri) const</td>
<td>find first archive which contains the file path</td>
</tr>
<tr>
<td><strong>Ptr&lt; Archive &gt;</strong> <strong>FindArchiveWithDir</strong> (const <strong>URI</strong> &amp;dirUri) const</td>
<td>find first archive which contains the directory path</td>
</tr>
</tbody>
</table>
void IO::ZipFileSystem::Setup() [inherited]

setup the archive file system

Setup the ZipFileSystem. Registers the ZipFileStream class.

Ptr< Archive > IO::ZipFileSystem::FindArchiveWithFile(const URI & uri ) const [inherited]

find first archive which contains the file path

This method takes a normal file URI and checks if the local path of the URI is contained as file entry in any mounted zip archive. If yes ptr to the zip archive is returned, otherwise a 0 pointer. NOTE: if the same path resides in several zip archives, it is currently not defined which one will be returned (the current implementation returns the first zip archive in alphabetical order which contains the file).

Ptr< Archive > IO::ZipFileSystem::FindArchiveWithDir(const URI & uri ) const [inherited]

find first archive which contains the directory path

Same as FindArchiveWithFile(), but checks for a directory entry in a zip file.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:46 2010
IO::Assign
#include <assign.h>

Inheritance diagram for IO::Assign:

Util::KeyValuePair< Util::String, Util::String >

IO::Assign

< Util::String, Util::String >" shape="rect" coords="0,0,258,24"
Detailed Description

An assign associates an intuitive name with a real-world filesystem path.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Assign ()</strong></td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><strong>Assign</strong> (const Util::String &amp;name, const Util::String &amp;path)</td>
<td><em>constructor with assign name and path</em></td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetName ()</strong> const</td>
<td><em>get name of assign</em></td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetPath ()</strong> const</td>
<td><em>get path of assign</em></td>
</tr>
<tr>
<td>bool <strong>operator==</strong> (const KeyValuePair&lt; Util::String, Util::String &gt; &amp;rhs) const</td>
<td><em>equality operator</em></td>
</tr>
<tr>
<td>bool <strong>operator!=</strong> (const KeyValuePair&lt; Util::String, Util::String &gt; &amp;rhs) const</td>
<td><em>inequality operator</em></td>
</tr>
<tr>
<td>bool <strong>operator&gt;</strong> (const KeyValuePair&lt; Util::String, Util::String &gt; &amp;rhs) const</td>
<td><em>greater operator</em></td>
</tr>
<tr>
<td>bool <strong>operator&gt;=</strong> (const KeyValuePair&lt; Util::String, Util::String &gt; &amp;rhs) const</td>
<td><em>greater-or-equal operator</em></td>
</tr>
<tr>
<td>bool <strong>operator&lt;</strong> (const KeyValuePair&lt; Util::String, Util::String &gt; &amp;rhs) const</td>
<td><em>lesser operator</em></td>
</tr>
<tr>
<td>bool <strong>operator&lt;=</strong> (const KeyValuePair&lt; Util::String, Util::String &gt; &amp;rhs) const</td>
<td><em>lesser-or-equal operator</em></td>
</tr>
<tr>
<td>Util::String &amp; <strong>Value ()</strong></td>
<td><em>read/write access to value</em></td>
</tr>
<tr>
<td>const Util::String &amp; <strong>Value ()</strong> const</td>
<td><em>read access to key</em></td>
</tr>
<tr>
<td>const Util::String &amp; <strong>Key ()</strong> const</td>
<td><em>read access to key</em></td>
</tr>
</tbody>
</table>
IO::<code>AssignRegistry</code>
#include <assignregistry.h>

Inheritance diagram for IO::AssignRegistry:
Detailed Description

Central registry for path assigns. This is a true singleton, visible from all threads.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AssignRegistry ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~AssignRegistry ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Setup ()</strong></td>
<td>setup the assign registry (may only be called once from the main thread)</td>
</tr>
<tr>
<td><strong>Discard ()</strong></td>
<td>discard the assign registry</td>
</tr>
<tr>
<td><strong>IsValid ()</strong> const</td>
<td>return true if the object has been setup</td>
</tr>
<tr>
<td><strong>SetAssign (const Assign &amp;assign)</strong></td>
<td>set a new assign</td>
</tr>
<tr>
<td><strong>HasAssign (const Util::String &amp;assignName)</strong></td>
<td>return true if an assign exists</td>
</tr>
<tr>
<td><strong>GetAssign (const Util::String &amp;assignName)</strong></td>
<td>get an assign</td>
</tr>
<tr>
<td><strong>ClearAssign (const Util::String &amp;assignName)</strong></td>
<td>clear an assign</td>
</tr>
<tr>
<td><strong>GetAllAssigns ()</strong> const</td>
<td>return an array of all currently defined assigns</td>
</tr>
<tr>
<td><strong>ResolveAssigns (const URI &amp;uri)</strong></td>
<td>resolve any assigns in an <strong>URI</strong></td>
</tr>
<tr>
<td><strong>ResolveAssignsInString (const Util::String &amp;uriString)</strong></td>
<td>resolve any assigns in a string (must have <strong>URI</strong> form)</td>
</tr>
<tr>
<td><strong>GetRefCount ()</strong> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

**URI**

```cpp
IO::AssignRegistry::ResolveAssigns (const URI& uri) const
```

resolve any assigns in an **URI**

Resolves all assigns from an **URI** returning an **URI**. It is allowed to "stack" assigns, which means, defining an assign as pointing to another assign.

**String**

```cpp
IO::AssignRegistry::ResolveAssignsInString (const Util::String& uriString) const
```

resolve any assigns in a string (must have **URI** form)

Resolves all assigns from a **URI**. It is allowed to "stack" assigns, which means, defining an assign as pointing to another assign.

**int**

```cpp
Core::RefCounted::GetRefCount () const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

**void**

```cpp
Core::RefCounted::AddRef () [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

**void**

```cpp
Core::RefCounted::Release () [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const `Util::String` &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

`Util::FourCC`
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

`void`
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
IO::BinaryReader
#include <binaryreader.h>

Inheritance diagram for IO::BinaryReader:
Detailed Description

A friendly interface to read binary data from a stream. Optionally the reader can use memory mapping for optimal read performance. Performs automatic byte order conversion if necessary.

Todo:
    convert endianess!

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BinaryReader ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~BinaryReader ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <code>SetMemoryMappingEnabled (bool b)</code></td>
<td>Call before <code>Open()</code> to enable memory mapping (if stream supports mapping)</td>
</tr>
<tr>
<td>bool <code>IsMemoryMappingEnabled ()</code> const</td>
<td>Return true if memory mapping is enabled</td>
</tr>
<tr>
<td>void <code>SetStreamByteOrder (System::ByteOrder::Type byteOrder)</code></td>
<td>Set the stream byte order (default is host byte order)</td>
</tr>
<tr>
<td>System::ByteOrder::Type <code>GetStreamByteOrder ()</code> const</td>
<td>Get the stream byte order</td>
</tr>
<tr>
<td>virtual bool <code>Open ()</code></td>
<td>Begin reading from the stream</td>
</tr>
<tr>
<td>virtual void <code>Close ()</code></td>
<td>End reading from the stream</td>
</tr>
<tr>
<td>char <code>ReadChar ()</code></td>
<td>Read an 8-bit char from the stream</td>
</tr>
<tr>
<td>unsigned char <code>ReadUChar ()</code></td>
<td>Read an 8-bit unsigned character from the stream</td>
</tr>
<tr>
<td>short <code>ReadShort ()</code></td>
<td>Read a 16-bit short from the stream</td>
</tr>
<tr>
<td>unsigned short <code>ReadUShort ()</code></td>
<td>Read a 16-bit unsigned short from the stream</td>
</tr>
<tr>
<td>int <code>ReadInt ()</code></td>
<td>Read a 32-bit int from the stream</td>
</tr>
<tr>
<td>unsigned int <code>ReadUInt ()</code></td>
<td>Read a 32-bit unsigned int from the stream</td>
</tr>
<tr>
<td>float <code>ReadFloat ()</code></td>
<td>Read a float value from the stream</td>
</tr>
<tr>
<td>float <code>ReadFloatFromNormalizedUByte2 ()</code></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Method</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>float</td>
<td><code>ReadFloatFromUnsignedNormalizedUByte2()</code></td>
</tr>
<tr>
<td>double</td>
<td><code>ReadDouble()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>ReadBool()</code></td>
</tr>
<tr>
<td>Util::String</td>
<td><code>ReadString()</code></td>
</tr>
<tr>
<td>Math::float2</td>
<td><code>ReadFloat2()</code></td>
</tr>
<tr>
<td>Math::float4</td>
<td><code>ReadFloat4()</code></td>
</tr>
<tr>
<td>Math::point</td>
<td><code>ReadPoint()</code></td>
</tr>
<tr>
<td>Math::vector</td>
<td><code>ReadVector()</code></td>
</tr>
<tr>
<td>Math::matrix44</td>
<td><code>ReadMatrix44()</code></td>
</tr>
<tr>
<td>Util::Guid</td>
<td><code>ReadGuid()</code></td>
</tr>
<tr>
<td>Util::Blob</td>
<td><code>ReadBlob()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>ReadRawData (void* ptr, SizeT numBytes)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetStream (const Ptr&lt; Stream &gt; &amp;s)</code></td>
</tr>
<tr>
<td>const Ptr&lt; Stream &gt; &amp;</td>
<td><code>GetStream()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>HasStream()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>Eof()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsOpen()</code></td>
</tr>
</tbody>
</table>
return true if currently open

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void IO::StreamReader::SetStream(const Ptr<Stream> & s) [inherited]
```

set stream to read from

Attaches the reader to a stream. This will increment the refcount of the stream.

Reimplemented in Messaging::MessageReader.

```cpp
const Ptr<Stream> & IO::StreamReader::GetStream() const [inherited]
```

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use HasStream() to determine if a stream is attached.

```cpp
bool IO::StreamReader::HasStream() const [inherited]
```

return true if a stream is set

Returns true if a stream is attached to the reader.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
IO::BinaryWriter
IO::BinaryWriter Class Reference

#include <binarywriter.h>

Inheritance diagram for IO::BinaryWriter:
Detailed Description

A friendly interface for writing binary data to a stream. Optionally the writer can use memory mapping for optimal write performance.

Todo:
  convert endianess!

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BinaryWriter ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~BinaryWriter ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>void SetMemoryMappingEnabled (bool b)</td>
<td>Call before Open() to enable memory mapping (if stream supports mapping)</td>
</tr>
<tr>
<td>bool IsMemoryMappingEnabled () const</td>
<td>Return true if memory mapping is enabled</td>
</tr>
<tr>
<td>void SetStreamByteOrder (System::ByteOrder::Type byteOrder)</td>
<td>Set the stream byte order (default is host byte order)</td>
</tr>
<tr>
<td>System::ByteOrder::Type GetStreamByteOrder () const</td>
<td>Get the stream byte order</td>
</tr>
<tr>
<td>virtual bool Open ()</td>
<td>Begin reading from the stream</td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td>End reading from the stream</td>
</tr>
<tr>
<td>void WriteChar (char c)</td>
<td>Write an 8-bit char to the stream</td>
</tr>
<tr>
<td>void WriteUChar (unsigned char c)</td>
<td>Write an 8-bit unsigned char to the stream</td>
</tr>
<tr>
<td>void WriteShort (short s)</td>
<td>Write an 16-bit short to the stream</td>
</tr>
<tr>
<td>void WriteUShort (unsigned short s)</td>
<td>Write an 16-bit unsigned short to the stream</td>
</tr>
<tr>
<td>void WriteInt (int i)</td>
<td>Write an 32-bit int to the stream</td>
</tr>
<tr>
<td>void WriteUInt (unsigned int i)</td>
<td>Write an 32-bit unsigned int to the stream</td>
</tr>
<tr>
<td>void WriteFloat (float f)</td>
<td>Write a float value to the stream</td>
</tr>
<tr>
<td>void WriteFloatAsNormalizedUByte2 (float f)</td>
<td></td>
</tr>
</tbody>
</table>
void **WriteFloatAsUnsignedNormalizedUByte2** (float f)
write a compressed float value to the stream, lossy and needed to be in the range of -1.0 and +1.0

void **WriteDouble** (double d)
write a double value to the stream

void **WriteBool** (bool b)
write a boolean value to the stream

void **WriteString** (const Util::String &s)
write a string to the stream

void **WriteFloat2** (Math::float2 f)
write a float value to the stream

void **WriteFloat4** (const Math::float4 &v)
write a float4 to the stream

void **WritePoint** (const Math::point &v)
write a float4 to the stream

void **WriteVector** (const Math::vector &v)
write a float4 to the stream

void **WriteMatrix44** (const Math::matrix44 &m)
write a matrix44 to the stream

void **WriteGuid** (const Util::Guid &guid)
write a guid

void **WriteBlob** (const Util::Blob &blob)
write a blob of data

void **WriteRawData** (const void *ptr, SizeT numBytes)
write raw data

void **SetStream** (const Ptr< Stream > &s)
set stream to write to

const Ptr< Stream > & **GetStream** () const
get currently set stream

bool **HasStream** () const
return true if a stream is set

bool **IsOpen** () const
return true if currently open
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetRefCount()</code></td>
<td>const get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>const return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code></td>
<td>const return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>const return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti)</code></td>
<td>const return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName)</code></td>
<td>const return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>const return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName()</code></td>
<td>const get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC()</code></td>
<td>const get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void IO::BinaryWriter::WriteString(const Util::String & s)
```
write a string to the stream

NOTE: for strings, first the length will be written into a 32-bit int, then the string contents without the 0-terminator.

```cpp
void IO::StreamWriter::SetStream(const Ptr<Stream> & s) [inherited]
```
set stream to write to

Attaches the writer to a stream. This will increment the refcount of the stream.

Reimplemented in Messaging::MessageWriter.

```cpp
const Ptr<Stream> & IO::StreamWriter::GetStream() const [inherited]
```
get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use HasStream() to determine if a stream is attached.

```cpp
bool IO::StreamWriter::HasStream() const [inherited]
```
return true if a stream is set

Returns true if a stream is attached to the writer.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

    void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

    void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

    const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

    Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

    void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
IO::BXmLoaderUtil
IO::BXmlLoaderUtil Class Reference

#include <bxmlloaderutil.h>
Detailed Description

Helper class for loading binary xml files created with N3's binaryxmlconverter3 tool. BXmlLoaderUtil objects are usually not used directly, but are wrapped by higher level classes like XmlReader.

Uses raw char pointers so that the class can both be used with Nebula2 and Nebula3.

(C) 2009 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BXmlLoaderUtil ()</td>
<td>constructor</td>
</tr>
<tr>
<td>~BXmlLoaderUtil ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void SetupFromFileInMemory (void *buf, SizeT size)</td>
<td>setup the object from a file-in-memory, buffer must remain intact until Discard() is called!</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>discard the object</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>return true if the object has been setup</td>
</tr>
<tr>
<td>ushort FindNodeIndex (const char *path) const</td>
<td>find a node index by path</td>
</tr>
<tr>
<td>void SetCurrentNodeIndex (ushort index)</td>
<td>set cursor to node index</td>
</tr>
<tr>
<td>ushort GetCurrentNodeIndex () const</td>
<td>get current node index</td>
</tr>
<tr>
<td>const char * GetCurrentNodeName () const</td>
<td>get name of current node</td>
</tr>
<tr>
<td>bool SetToFirstChild (const char *name)</td>
<td>set cursor to first matching child node, name may point to empty string</td>
</tr>
<tr>
<td>bool SetToNextChild (const char *name)</td>
<td>set cursor to next matching child node, name may point to empty string</td>
</tr>
<tr>
<td>bool SetToParent ()</td>
<td>set to parent node</td>
</tr>
<tr>
<td>uint FindAttrIndex (const char *attrName)</td>
<td>find attribute index by name</td>
</tr>
<tr>
<td>uint GetNumAttrs () const</td>
<td>get number of attributes on node</td>
</tr>
<tr>
<td>const char * GetAttrName (uint attrIndex)</td>
<td>get name of attribute by index</td>
</tr>
<tr>
<td>const char * GetAttrValue (uint attrIndex)</td>
<td>get value of attribute by index</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>ushort FindChildNodeIndex(ushort nodeIndex, const char *childName) const</code></td>
<td>find a child node index by name, name may point to empty string</td>
</tr>
<tr>
<td><code>ushort FindSiblingNodeIndex(ushort nodeIndex, const char *sibName) const</code></td>
<td>find a sibling node index by name, name may point to empty string</td>
</tr>
<tr>
<td><code>ushort GetParentNodeIndex(ushort nodeIndex) const</code></td>
<td>get the parent node index of a node</td>
</tr>
<tr>
<td><code>const char * GetNodeName(ushort nodeIndex) const</code></td>
<td>get name of a given node</td>
</tr>
</tbody>
</table>
Member Function Documentation

uint
IO::BXmlLoaderUtil::FindAttrIndex ( const char * attrName ) const

find attribute index by name

Returns a local attribute index by name, this is NOT an index into the
global attributes array!

ushort
IO::BXmlLoaderUtil::FindChildNodeIndex ( ushort nodeIndex,
const char * childName
) const

find a child node index by name, name may point to empty string

NOTE: childName may be 0 or point to an empty string

ushort
IO::BXmlLoaderUtil::FindSiblingNodeIndex ( ushort nodeIndex,
const char * sibName
) const

find a sibling node index by name, name may point to empty string

NOTE: childName may be 0 or point to an empty string

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:46 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::BXmlReader
#include <bxmlreader.h>

Inheritance diagram for IO::BXmlReader:
Detailed Description

**Stream** reader for binary XML files. The interface is similar to **XmlReader** so that it is easy to switch between the 2 classes. Use the N3 tool binaryxmlconverter3.exe to convert an XML file to a binary XML file.

(C) 2009 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BXmlReader ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~BXmlReader ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual bool Open ()</td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td>end reading from the stream</td>
</tr>
<tr>
<td>bool HasNode (const Util::String &amp;path)</td>
<td>return true if node exists</td>
</tr>
<tr>
<td>Util::String GetCurrentNodeName ()</td>
<td>get short name of current node</td>
</tr>
<tr>
<td>Util::String GetCurrentNodePath ()</td>
<td>get path to current node</td>
</tr>
<tr>
<td>void SetToNode (const Util::String &amp;path)</td>
<td>set current node as path</td>
</tr>
<tr>
<td>bool SetToFirstChild (const Util::String &amp;name=&quot;&quot;)</td>
<td>set current node to first child node, return false if no child exists</td>
</tr>
<tr>
<td>bool SetToNextChild (const Util::String &amp;name=&quot;&quot;)</td>
<td>set current node to next sibling node, return false if no more sibling exists</td>
</tr>
<tr>
<td>bool SetToParent ()</td>
<td>set current node to parent, return false if no parent exists</td>
</tr>
<tr>
<td>bool HasAttr (const char *attr) const</td>
<td>return true if matching attribute exists on current node</td>
</tr>
<tr>
<td>Util::Array&lt; Util::String &gt; GetAttrs () const</td>
<td>return names of all attrs on current node</td>
</tr>
<tr>
<td>Util::String GetString (const char *attr) const</td>
<td>get string attribute value from current node</td>
</tr>
<tr>
<td>bool GetBool (const char *attr) const</td>
<td>get bool attribute value from current node</td>
</tr>
<tr>
<td>int GetInt (const char *attr) const</td>
<td>get int attribute value from current node</td>
</tr>
<tr>
<td>Type</td>
<td>Method Name</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>float</td>
<td><code>GetFloat</code></td>
</tr>
<tr>
<td><code>Math::float2</code></td>
<td><code>GetFloat2</code></td>
</tr>
<tr>
<td><code>Math::float4</code></td>
<td><code>GetFloat4</code></td>
</tr>
<tr>
<td><code>Math::matrix44</code></td>
<td><code>GetMatrix44</code></td>
</tr>
<tr>
<td>template&lt;typename T&gt;</td>
<td><code>Get</code></td>
</tr>
<tr>
<td><code>Util::String</code></td>
<td><code>GetOptString</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>GetOptBool</code></td>
</tr>
<tr>
<td>int</td>
<td><code>GetOptInt</code></td>
</tr>
<tr>
<td>float</td>
<td><code>GetOptFloat</code></td>
</tr>
<tr>
<td><code>Math::float2</code></td>
<td><code>GetOptFloat2</code></td>
</tr>
<tr>
<td><code>Math::float4</code></td>
<td><code>GetOptFloat4</code></td>
</tr>
<tr>
<td><code>Math::matrix44</code></td>
<td><code>GetOptMatrix44</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetStream</code></td>
</tr>
<tr>
<td>const <code>Ptr&lt; Stream &gt;</code> &amp;</td>
<td><code>GetStream</code></td>
</tr>
<tr>
<td>bool</td>
<td></td>
</tr>
<tr>
<td>Member Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>HasStream()</code> const</td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td><code>Eof()</code> const</td>
<td>return true if the stream has reached EOF</td>
</tr>
<tr>
<td><code>IsOpen()</code> const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td><code>GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName()</code> const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC()</code> const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void IO::StreamReader::SetStream(const Ptr<Stream> & s) [inherited]
```

set stream to read from

Attaches the reader to a stream. This will increment the refcount of the stream.

Reimplemented in `Messaging::MessageReader`.

```cpp
const Ptr<Stream> & IO::StreamReader::GetStream() const [inherited]
```

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use `HasStream()` to determine if a stream is attached.

```cpp
bool IO::StreamReader::HasStream() const [inherited]
```

return true if a stream is set

Returns true if a stream is attached to the reader.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
IO::Console
#include <console.h>

Inheritance diagram for IO::Console:

```
Core::RefCounted

IO::Console
```

IO::Console Class Reference
Detailed Description

Nebula3's console, this is the central place for command-line-style communication with the user. By default, all output will just disappear unless console handlers are added. **Console** handlers are user-derivable objects which do something with the output and may provide text input.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Console ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~Console ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Open ()</strong></td>
<td>open the console</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close the console</td>
</tr>
<tr>
<td><strong>bool IsOpen () const</strong></td>
<td>return true if currently open</td>
</tr>
<tr>
<td><strong>void Update ()</strong></td>
<td>called per-frame</td>
</tr>
<tr>
<td><strong>void AttachHandler (const Ptr&lt;ConsoleHandler&gt; &amp;handler)</strong></td>
<td>attach a console handler to the console</td>
</tr>
<tr>
<td><strong>void RemoveHandler (const Ptr&lt;ConsoleHandler&gt; &amp;handler)</strong></td>
<td>remove a console handler from the console</td>
</tr>
<tr>
<td><strong>Util::Array&lt; Ptr&lt;ConsoleHandler&gt; &gt; GetHandlers () const</strong></td>
<td>get array of currently installed handlers</td>
</tr>
<tr>
<td><strong>bool HasInput () const</strong></td>
<td>return true if user input is available</td>
</tr>
<tr>
<td><strong>Util::String GetInput () const</strong></td>
<td>get user input</td>
</tr>
<tr>
<td>*<em>void __cdecl Print (const char <em>fmt, ...)</em></em></td>
<td>print a formatted line (printf style)</td>
</tr>
<tr>
<td>*<em>void __cdecl Print (const char <em>fmt, va_list argList)</em></em></td>
<td>print a formatted line (printf style)</td>
</tr>
<tr>
<td><strong>void Print (const Util::String &amp;s)</strong></td>
<td>print a string object</td>
</tr>
<tr>
<td>*<em>void __cdecl Error (const char <em>fmt, ...)</em></em></td>
<td>put an error message and cancel execution</td>
</tr>
<tr>
<td>*<em>void __cdecl Error (const char <em>fmt, va_list argList)</em></em></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Warning (const char *fmt,...)</td>
<td>put an error message and cancel execution</td>
</tr>
<tr>
<td>Warning (const char *fmt, va_list argList)</td>
<td>put a warning message</td>
</tr>
<tr>
<td>Confirm (const char *fmt,...)</td>
<td>display a confirmation message box</td>
</tr>
<tr>
<td>Confirm (const char *fmt, va_list argList)</td>
<td>display a confirmation message box</td>
</tr>
<tr>
<td>DebugOut (const char *fmt,...)</td>
<td>print a debug-only message</td>
</tr>
<tr>
<td>DebugOut (const char *fmt, va_list argList)</td>
<td>print a debug-only message</td>
</tr>
<tr>
<td>GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>IsA (const Util::String &amp;rttiName) const</td>
<td></td>
</tr>
</tbody>
</table>
bool IsA (const Util::FourCC &rttiFourCC) const

return true if this object is instance of given class, or a derived class, by string

const Util::String & GetClassName () const

get the class name

Util::FourCC GetClassFourCC () const

get the class FourCC code
<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void IO::Console::Update()
called per-frame

This method may only be called from the main thread!

int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
IO::ConsoleHandler
#include <consolehandler.h>

Inheritance diagram for IO::ConsoleHandler:
Detailed Description

**Base** class for all console handlers. **Console** handlers are attached to Nebula3's central console object and are notified by the console object about output and deliver input to the console.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConsoleHandler</strong></td>
<td>() virtual constructor</td>
</tr>
<tr>
<td>~<strong>ConsoleHandler</strong></td>
<td>() virtual destructor</td>
</tr>
<tr>
<td>virtual void <strong>Open</strong></td>
<td>() called by console when attached</td>
</tr>
<tr>
<td>virtual void <strong>Close</strong></td>
<td>() called by console when removed</td>
</tr>
<tr>
<td>bool <strong>IsOpen</strong></td>
<td>() const return true if currently open</td>
</tr>
<tr>
<td>virtual void <strong>Update</strong></td>
<td>() called by Console::Update()</td>
</tr>
<tr>
<td>virtual void <strong>Print</strong></td>
<td>(const Util::String &amp;s) called by console to output data</td>
</tr>
<tr>
<td>virtual void <strong>Error</strong></td>
<td>(const Util::String &amp;s) called by console with serious error</td>
</tr>
<tr>
<td>virtual void <strong>Warning</strong></td>
<td>(const Util::String &amp;s) called by console to output warning</td>
</tr>
<tr>
<td>virtual void <strong>Confirm</strong></td>
<td>(const Util::String &amp;s) called by console to display a confirmation message box</td>
</tr>
<tr>
<td>virtual void <strong>DebugOut</strong></td>
<td>(const Util::String &amp;s) called by console to output debug string</td>
</tr>
<tr>
<td>bool <strong>HasInput</strong></td>
<td>()虚 return true if input is available</td>
</tr>
<tr>
<td>virtual Util::String <strong>GetInput</strong></td>
<td>() read available input</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong></td>
<td>() const get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong></td>
<td>() increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong></td>
<td>() decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong></td>
<td>(const Rtti &amp;rtti) const</td>
</tr>
</tbody>
</table>
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
Static Public Member Functions

| static void DumpRefCountingLeaks () | dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!) |
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::ExcelXmlReader
#include <excelxmlreader.h>

Inheritance diagram for IO::ExcelXmlReader:
Detailed Description

A stream reader class which reads Excel XML files. NOTE: the strings returned by this class will be in UTF-8 format!

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ExcelXmlReader</strong> (constructor)</td>
<td>Public constructor for ExcelXmlReader</td>
</tr>
<tr>
<td>virtual ~ExcelXmlReader (destructor)</td>
<td>Public destructor for ExcelXmlReader</td>
</tr>
<tr>
<td>virtual bool Open (begin reading from the stream)</td>
<td>Public method to open the stream and begin reading from the stream</td>
</tr>
<tr>
<td>virtual void Close (end reading from the stream)</td>
<td>Public method to close the stream and end reading from the stream</td>
</tr>
<tr>
<td>SizeT GetNumTables (const) get the number of tables in the file</td>
<td>Public method to get the number of tables in the file</td>
</tr>
<tr>
<td>const Util::String &amp; GetTableName (IndexT tableIndex=0) get the name of the table at given table index</td>
<td>Public method to get the name of the table at a given table index</td>
</tr>
<tr>
<td>IndexT GetTableIndex (const Util::String &amp;tableName) get index of table by name</td>
<td>Public method to get the index of a table by its name</td>
</tr>
<tr>
<td>SizeT GetNumRows (IndexT tableIndex=0) get number of rows in table</td>
<td>Public method to get the number of rows in a table</td>
</tr>
<tr>
<td>SizeT GetNumColumns (IndexT tableIndex=0) get number of columns in table</td>
<td>Public method to get the number of columns in a table</td>
</tr>
<tr>
<td>bool HasColumn (const Util::String &amp;columnName, IndexT tableIndex=0) return true if table has named column</td>
<td>Public method to check if a table has a named column</td>
</tr>
<tr>
<td>IndexT FindColumnIndex (const Util::String &amp;columnName, IndexT tableIndex=0) get column index by name, returns InvalidIndex if column doesn't exist</td>
<td>Public method to get the index of a column by its name</td>
</tr>
<tr>
<td>const Util::String &amp; GetElement (IndexT rowIndex, IndexT columnIndex, IndexT tableIndex=0) get cell content by row index and column index</td>
<td>Public method to get the cell content by row index and column index</td>
</tr>
<tr>
<td>const Util::String &amp; GetElement (IndexT rowIndex, const Util::String &amp;columnName, IndexT tableIndex=0) get cell content by row index and column name (SLOW!)</td>
<td>Public method to get the cell content by row index and column name (SLOW!)</td>
</tr>
<tr>
<td>void SetStream (const Ptr&lt; Stream &gt; &amp;) set stream to read from</td>
<td>Public method to set the stream to read from</td>
</tr>
<tr>
<td>const Ptr&lt; Stream &gt; &amp;</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetStream()</code> const</td>
<td>get currently set stream</td>
</tr>
<tr>
<td><code>HasStream()</code> const</td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td><code>Eof()</code> const</td>
<td>return true if the stream has reached EOF</td>
</tr>
<tr>
<td><code>IsOpen()</code> const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td><code>GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName()</code> const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC()</code> const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool IO::ExcelXmlReader::Open() [virtual]
begin reading from the stream
Open the Excel-XML-stream and completely parse its content.
Reimplemented from IO::StreamReader.
```

```cpp
bool IO::ExcelXmlReader::HasColumn(const Util::String columnName, 
                                  const IndexT tableIndex = 0)
return true if table has named column
NOTE: this method is slow because it does a linear search over the column names.
```

```cpp
IndexT IO::ExcelXmlReader::FindColumnIndex(const Util::String columnName, 
                                         const IndexT tableIndex = 0)
get column index by name, returns InvalidIndex if column doesn't exist
NOTE: this method is slow because it does a linear search over the column names.
```

```cpp
void IO::StreamReader::SetStream(const Ptr<Stream> & s) [inherited]
set stream to read from
```
Attaches the reader to a stream. This will increment the refcount of the stream.

Reimplemented in **Messaging::MessageReader**.

```cpp
class IO::StreamReader::GetStream()
```

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use **HasStream()** to determine if a stream is attached.

```cpp
bool IO::StreamReader::HasStream()
```

return true if a stream is set

Returns true if a stream is attached to the reader.

```cpp
int Core::RefCounted::GetRefCount()
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef()
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release()
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const `Util::String` &
Core::RefCounted::GetClassName

get the class name

Get the class name of the object.

`Util::FourCC`
Core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
IO::FileStream
IO::FileStream Class Reference

#include <filestream.h>

Inheritance diagram for IO::FileStream:
Detailed Description

A stream to which offers read/write access to filesystem files.

(C) 2006 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enum</code></td>
<td><strong>AccessMode</strong></td>
</tr>
<tr>
<td></td>
<td>access modes</td>
</tr>
<tr>
<td><code>enum</code></td>
<td><strong>AccessPattern</strong></td>
</tr>
<tr>
<td></td>
<td>access prefered pattern</td>
</tr>
<tr>
<td><code>enum</code></td>
<td><strong>SeekOrigin</strong></td>
</tr>
<tr>
<td></td>
<td>seek origins</td>
</tr>
<tr>
<td><code>typedef int</code></td>
<td><strong>Position</strong></td>
</tr>
<tr>
<td></td>
<td>typedefs</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FileStream ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~FileStream ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>CanRead () const</strong></td>
<td>supports reading?</td>
</tr>
<tr>
<td><strong>CanWrite () const</strong></td>
<td>supports writing?</td>
</tr>
<tr>
<td><strong>CanSeek () const</strong></td>
<td>supports seeking?</td>
</tr>
<tr>
<td><strong>CanBeMapped () const</strong></td>
<td>supports memory mapping (read-only)?</td>
</tr>
<tr>
<td><strong>GetSize () const</strong></td>
<td>get the size of the stream in bytes</td>
</tr>
<tr>
<td><strong>GetPosition () const</strong></td>
<td>get the current position of the read/write cursor</td>
</tr>
<tr>
<td><strong>Open ()</strong></td>
<td>open the stream</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>close the stream</td>
</tr>
<tr>
<td>*<em>Write (const void <em>ptr, Size numBytes)</em></em></td>
<td>directly write to the stream</td>
</tr>
<tr>
<td>*<em>Read (void <em>ptr, Size numBytes)</em></em></td>
<td>directly read from the stream</td>
</tr>
<tr>
<td><strong>Seek (Offset offset, SeekOrigin origin)</strong></td>
<td>seek in stream</td>
</tr>
<tr>
<td><strong>Flush ()</strong></td>
<td>flush unsaved data</td>
</tr>
<tr>
<td><strong>Eof () const</strong></td>
<td>return true if end-of-stream reached</td>
</tr>
<tr>
<td><strong>Map ()</strong></td>
<td>map stream to memory</td>
</tr>
<tr>
<td><strong>Unmap ()</strong></td>
<td></td>
</tr>
</tbody>
</table>
void SetURI (const URI &u)  
set stream location as URI

const URI & GetURI () const 
get stream URI

virtual void SetSize (Size s) 
set a new size for the stream

void SetAccessMode (AccessMode m) 
set the access mode of the stream (default is ReadAccess)

AccessMode GetAccessMode () const 
get the access mode of the stream

void SetAccessPattern (AccessPattern p) 
set the preferred access pattern (default is Sequential)

AccessPattern GetAccessPattern () const 
get the preferred access pattern

void SetMediaType (const MediaType &t) 
set optional media type of stream content

const MediaType & GetMediaType () const 
get optional media type

bool isOpen () const 
return true if currently open

virtual void Write (const void *ptr, Size numBytes) 
directly write to the stream

virtual Size Read (void *ptr, Size numBytes) 
directly read from the stream

virtual void Seek (Offset offset, SeekOrigin origin) 
seek in stream

bool IsMapped () const 
return true if stream is currently mapped to memory

int GetRefCount () const 
get the current refcount

void AddRef () 
increment refcount by one

void Release () 
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const 
return true if this object is instance of given class
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void DumpRefCountingLeaks ()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool
IO::FileStream::CanBeMapped() const [virtual]
supports memory mapping (read-only)?
FileStreams support mapping (only read access for now).
Reimplemented from IO::Stream.

void
IO::Stream::SetURI(const URI & u) [inherited]
set stream location as URI
Set the URI of the stream as string. The URI identifies the source resource of the stream.

const URI &
IO::Stream::GetURI() const [inherited]
get stream URI
Get the URI of the stream as string.

void
IO::Stream::SetSize(Size s) [virtual, inherited]
set a new size for the stream
This sets a new size for the stream. Not all streams support this method. If the new size if smaller then the existing size, the contents will be clipped.

void
IO::Stream::SetAccessMode(AccessMode m) [inherited]
set the access mode of the stream (default is ReadAccess)
This method sets the intended access mode of the stream. The actual behaviour depends on the implementation of the derived class. The default is ReadWrite.

```
Stream::AccessMode
IO::Stream::GetAccessMode() const [inherited]
```

get the access mode of the stream

Get the access mode of the stream.

```
void
IO::Stream::SetAccessPattern(AccessPattern p) [inherited]
```

set the preferred access pattern (default is Sequential)

Set the preferred access pattern of the stream. This can be Random or Sequential. This is an optional flag to improve performance with some stream implementations. The default is sequential. The pattern cannot be changed while the stream is open.

```
Stream::AccessPattern
IO::Stream::GetAccessPattern() const [inherited]
```

get the preferred access pattern

Get the currently set preferred access pattern of the stream.

```
bool
IO::Stream::IsOpen() const [inherited]
```

return true if currently open

Return true if the stream is currently open.

```
void
IO::Stream::Write(const void* ptr, Size numBytes)
)[virtual, inherited]
```

directly write to the stream
Write raw data to the stream. For more convenient writing, attach the stream to an **IO::StreamWriter** object. This method is only valid if the stream class returns true in **CanWrite()**.

```cpp
Stream::Size IO::Stream::Read ( *ptr, 
    Size numBytes 
) [virtual, inherited]
```

directly read from the stream

Read raw data from the stream. For more convenient reading, attach the stream to an **IO::StreamReader** object. The method returns the number of bytes actually read. This method is only valid if the stream class returns true in **CanRead()**. Returns the number of bytes actually read from the stream, this may be less then numBytes, or 0 if end-of-stream is reached.

```cpp
void IO::Stream::Seek ( Offset offset, 
    SeekOrigin origin 
) [virtual, inherited]
```

seek in stream

Move the read/write cursor to a new position, returns the new position in the stream. This method is only supported if the stream class returns true in **CanSeek()**.

```cpp
bool IO::Stream::IsMapped ( ) const [inherited]
```

return true if stream is currently mapped to memory

Returns true if the stream is currently mapped.

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::FileTime
IO::FileTime Class Reference

#include <filetime.h>
Detailed Description

Defines a file-access timestamp.

(C) 2006 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::FSWrapper
#include <fswrapper.h>

Inheritance diagram for IO::FSWrapper:
Detailed Description

This is an internal IO class used to wrap platform specific filesystem access into a generic class. To port the filesystem code to a new platform all that has to be done is to write a new FSWrapper class.

(C) 2006 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::GameContentServer
#include <gamecontentserver.h>

Inheritance diagram for IO::GameContentServer:

```
  Core::RefCounted
    ↓
  Base::GameContentServerBase
    ↓
  IO::GameContentServer
```
Detailed Description

The game content server initializes access to game content on console platforms. The GameContentServer must be created by the main thread before the first IoServer is created.

(C) 2009 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GameContentServer()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~GameContentServer()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <code>SetTitle(const Util::String &amp;title)</code></td>
<td>Set human readable game title</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetTitle()</code> const</td>
<td>Get human readable game title</td>
</tr>
<tr>
<td>void <code>SetTitleId(const Util::String &amp;titleId)</code></td>
<td>Set title id</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetTitleId()</code> const</td>
<td>Get title id</td>
</tr>
<tr>
<td>void <code>SetTitleVersion(const Util::String &amp;version)</code></td>
<td>Set title version</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetVersion()</code> const</td>
<td>Get title version</td>
</tr>
<tr>
<td>void <code>Setup()</code></td>
<td>Setup the object</td>
</tr>
<tr>
<td>void <code>Discard()</code></td>
<td>Discard the object</td>
</tr>
<tr>
<td>bool <code>IsValid()</code></td>
<td>Return true if object has been setup</td>
</tr>
<tr>
<td>int <code>GetRefCount()</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf(const Util::String &amp;className)</code> const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetClassName</strong> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC <strong>GetClassFourCC</strong> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

int 
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void 
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void 
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void 
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
IO::HistoryConsoleHandler
IO::HistoryConsoleHandler Class Reference

#include <historyconsolehandler.h>

Inheritance diagram for IO::HistoryConsoleHandler:
Detailed Description

A console handler which stores the last N log messages in a Util::RingBuffer<String>.

(C) 2008 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HistoryConsoleHandler()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>void SetHistorySize(SizeT numLines)</code></td>
<td>Set history size</td>
</tr>
<tr>
<td><code>SizeT GetHistorySize()</code> const</td>
<td>Get history size</td>
</tr>
<tr>
<td><code>const Util::RingBuffer&lt; Util::String &gt;&amp; GetHistory()</code> const</td>
<td>Get accumulated log messages</td>
</tr>
<tr>
<td>virtual void <code>Print(const Util::String &amp;s)</code></td>
<td>Called by console to output data</td>
</tr>
<tr>
<td>virtual void <code>Error(const Util::String &amp;s)</code></td>
<td>Called by console with serious error</td>
</tr>
<tr>
<td>virtual void <code>Warning(const Util::String &amp;s)</code></td>
<td>Called by console to output warning</td>
</tr>
<tr>
<td>virtual void <code>DebugOut(const Util::String &amp;s)</code></td>
<td>Called by console to output debug string</td>
</tr>
<tr>
<td>virtual void <code>Open()</code></td>
<td>Called by console when attached</td>
</tr>
<tr>
<td>virtual void <code>Close()</code></td>
<td>Called by console when removed</td>
</tr>
<tr>
<td><code>bool IsOpen()</code></td>
<td>Return true if currently open</td>
</tr>
<tr>
<td>virtual void <code>Update()</code></td>
<td>Called by <code>Console::Update()</code></td>
</tr>
<tr>
<td>virtual void <code>Confirm(const Util::String &amp;s)</code></td>
<td>Called by console to display a confirmation message box</td>
</tr>
<tr>
<td>virtual bool <code>HasInput()</code></td>
<td>Return true if input is available</td>
</tr>
<tr>
<td><code>virtual Util::String GetInput()</code></td>
<td>Read available input</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName()</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC()</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
This method should be called as the very last before an application exits.
IO::Interface
IO::Interface Class Reference

#include <iointerface.h>
Detailed Description

Implements the asynchronous interface to the IO subsystem. This will run a minimal Nebula3 runtime with an IO subsystem in an extra thread. Communication with the IO::Interface happens by sending messages to the Interface object. Messages are guaranteed to be handled sequentially in FIFO order (there’s exactly one handler thread which handles all messages).

(C) 2006 Radon Labs GmbH
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**IO::interfaceHandler**
IO::IoInterfaceHandler Class Reference

#include <iointerfacehandler.h>

Inheritance diagram for IO::IoInterfaceHandler:
Detailed Description

Handler class for io interfaces.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IoInterfaceHandler ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~IoInterfaceHandler ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>Open ()</code></td>
<td>open the handler</td>
</tr>
<tr>
<td><code>Close ()</code></td>
<td>close the handler</td>
</tr>
<tr>
<td><code>HandleMessage (const Ptr&lt;Messaging::Message&gt; &amp;msg)</code></td>
<td>handle a message, return true if handled</td>
</tr>
<tr>
<td><code>SetCompanyName (const Util::StringAtom &amp;companyName)</code></td>
<td>set the company name</td>
</tr>
<tr>
<td><code>GetCompanyName () const</code></td>
<td>get the company name</td>
</tr>
<tr>
<td><code>SetAppName (const Util::StringAtom &amp;appName)</code></td>
<td>set the application name</td>
</tr>
<tr>
<td><code>GetAppName () const</code></td>
<td>get the application name</td>
</tr>
<tr>
<td><code>DoWork ()</code></td>
<td>optional &quot;per-frame&quot; DoWork method for continuous handlers</td>
</tr>
<tr>
<td><code>IsOpen () const</code></td>
<td>return true if open</td>
</tr>
<tr>
<td><code>GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsInstanceOf (const Util::String &amp;className)</code> const</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class by string</em></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsInstanceOf (const Util::FourCC &amp;classFourCC)</code> const</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class by fourcc</em></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA (const Rtti &amp;rtti)</code> const</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class</em></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA (const Util::String &amp;rttiName)</code> const</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by string</em></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC)</code> const</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by fourcc</em></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><code>GetClassName ()</code> const</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><code>GetClassFourCC ()</code> const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><code>DumpRefCountingLeaks()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</code></td>
<td></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OnCreateDirectory</code></td>
<td>(const <code>Ptr&lt; IO::CreateDirectory &gt; &amp;msg)</code> handle CreateDirectory message</td>
</tr>
<tr>
<td><code>OnDeleteDirectory</code></td>
<td>(const <code>Ptr&lt; IO::DeleteDirectory &gt; &amp;msg)</code> handle DeleteDirectory message</td>
</tr>
<tr>
<td><code>OnDeleteFile</code></td>
<td>(const <code>Ptr&lt; IO::DeleteFile &gt; &amp;msg)</code>     handle DeleteFile message</td>
</tr>
<tr>
<td><code>OnWriteStream</code></td>
<td>(const <code>Ptr&lt; IO::WriteStream &gt; &amp;msg)</code>    handle WriteStream message</td>
</tr>
<tr>
<td><code>OnReadStream</code></td>
<td>(const <code>Ptr&lt; IO::ReadStream &gt; &amp;msg)</code>     handle ReadStream message</td>
</tr>
<tr>
<td><code>OnCopyFile</code></td>
<td>(const <code>Ptr&lt; IO::CopyFile &gt; &amp;msg)</code>       handle CopyFile message</td>
</tr>
<tr>
<td><code>OnMountArchive</code></td>
<td>(const <code>Ptr&lt; IO::MountArchive &gt; &amp;msg)</code>   handle MountArchive message</td>
</tr>
</tbody>
</table>
void IO::IoInterfaceHandler::Open() [virtual]

open the handler

Opens the Interface message handler which does all the interesting stuff. This method already runs in the handler thread. The method initializes a minimal thread local Nebula3 runtime, just enough to handle the IO messages.

Reimplemented from Messaging::Handler.

void IO::IoInterfaceHandler::Close() [virtual]

close the handler

Closes the Interface message handler. This will shut down the minimal Nebula3 runtime, the method runs in the handler thread and is called just before the thread shuts down.

Reimplemented from Messaging::Handler.

bool IO::IoInterfaceHandler::HandleMessage(const Ptr<Messaging::Message> & msg) [virtual]

handle a message, return true if handled

Handles incoming messages. This method runs in the handler thread.

Reimplemented from Messaging::Handler.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount
Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
IO::IoServer
#include <ioserver.h>

Inheritance diagram for IO::IoServer:

```
Core::RefCounted

IO::IoServer
```
Detailed Description

The central server object of the IO subsystem offers the following services:

associate stream classes with URI schemes create the right stream object for a given URI transparant (ZIP) archive support path assign management global filesystem manipulation and query methods

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IoServer()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~IoServer</strong>()</td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>bool MountArchive(const URI &amp;uri)</strong></td>
<td>Mount a file archive (without archive file extension!)</td>
</tr>
<tr>
<td><strong>void UnmountArchive(const URI &amp;uri)</strong></td>
<td>Unmount a file archive (without archive file extension!)</td>
</tr>
<tr>
<td><strong>bool IsArchiveMounted(const URI &amp;uri)</strong> const</td>
<td>Return true if a archive is mounted (without archive file extension!)</td>
</tr>
<tr>
<td><strong>void SetArchiveFileSystemEnabled(bool b)</strong></td>
<td>Enable/disable transparent archive filesystem layering (default is yes)</td>
</tr>
<tr>
<td><strong>bool IsArchiveFileSystemEnabled()</strong> const</td>
<td>Return true if transparent archive filesystem is enabled</td>
</tr>
<tr>
<td><strong>void MountStandardArchives()</strong></td>
<td>Mount standard archives (e.g. home:export.zip and home:export_.zip)</td>
</tr>
<tr>
<td><strong>void UnmountStandardArchives()</strong></td>
<td>Unmount standard archives</td>
</tr>
<tr>
<td><strong>Ptr&lt;Stream&gt; CreateStream(const URI &amp;uri)</strong> const</td>
<td>Create a stream object for the given uri</td>
</tr>
<tr>
<td><strong>bool CreateDirectory(const URI &amp;uri)</strong> const</td>
<td>Create all missing directories in the path</td>
</tr>
<tr>
<td><strong>bool DeleteDirectory(const URI &amp;path)</strong> const</td>
<td>Delete an empty directory</td>
</tr>
<tr>
<td><strong>bool DirectoryExists(const URI &amp;path)</strong> const</td>
<td>Return true if directory exists</td>
</tr>
<tr>
<td><strong>bool CopyFile(const URI &amp;from, const URI &amp;to)</strong> const</td>
<td>Copy a file</td>
</tr>
<tr>
<td><strong>bool DeleteFile(const URI &amp;path)</strong> const</td>
<td>Delete a file</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool FileExists (const URI &amp;path) const</code></td>
<td>return true if file exists</td>
</tr>
<tr>
<td><code>void SetReadOnly (const URI &amp;path, bool b) const</code></td>
<td>set the readonly status of a file</td>
</tr>
<tr>
<td><code>bool IsReadOnly (const URI &amp;path) const</code></td>
<td>return read only status of a file</td>
</tr>
<tr>
<td><code>unsigned int ComputeFileCrc (const URI &amp;path) const</code></td>
<td>get the CRC checksum of a file</td>
</tr>
<tr>
<td><code>void SetFileWriteTime (const URI &amp;path, FileTime fileTime)</code></td>
<td>set the write-time of a file</td>
</tr>
<tr>
<td><code>FileTime GetFileWriteTime (const URI &amp;path) const</code></td>
<td>return the last write-time of a file</td>
</tr>
<tr>
<td><code>Util::Array&lt; Util::String &gt; ListFiles (const URI &amp;dir, const Util::String &amp;pattern, bool asFullPath=false) const</code></td>
<td>list all files matching a pattern in a directory</td>
</tr>
<tr>
<td><code>Util::Array&lt; Util::String &gt; ListDirectories (const URI &amp;dir, const Util::String &amp;pattern, bool asFullPath=false) const</code></td>
<td>list all subdirectories matching a pattern in a directory</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const</td>
<td><strong>Util::String</strong> &amp; <strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
</table>

*dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)*
Member Function Documentation

```cpp
void IO::IoServer::SetArchiveFileSystemEnabled ( bool b ) [inline]
```

enable/disable transparent archive filesystem layering (default is yes)

NOTE: on platforms which provide transparent archive access this method is point less (the archiveFileSystemEnabled flag will be ignored, and `IsArchiveFileSystemEnabled()` will always return false).

```cpp
bool IO::IoServer::IsArchiveFileSystemEnabled ( ) const [inline]
```

return true if transparent archive filesystem is enabled

NOTE: on platforms which provide transparent archive access through the OS (like on PS3) this method will always return false. This saves some unnecessary overhead in the Nebula3 `IoServer`.

```cpp
bool IO::IoServer::CreateDirectory ( const URI & uri ) const
```

create all missing directories in the path

This method creates all missing directories in a path.

```cpp
bool IO::IoServer::CopyFile ( const URI & fromUri, const URI & toUri ) const
```

copy a file

This copies a file to another file.
IO::IoServer::ComputeFileCrc (URI uri) const

get the CRC checksum of a file

This method computes the CRC checksum for a file.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC code of the object.
Core::RefCounted::DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::LogFileConsoleHandler
IO::LogFileConsoleHandler Class Reference

#include <logfileconsolehandler.h>

Inheritance diagram for IO::LogFileConsoleHandler:
Detailed Description

A console handler which writes all console output to a log file. The log file will be called appname_calendartime.txt and will be written into a bin:logfiles directory.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogFileConsoleHandler ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~LogFileConsoleHandler ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void Open ()</td>
<td>called by console when attached</td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td>called by console when removed</td>
</tr>
<tr>
<td>virtual void Print (const Util::String &amp;s)</td>
<td>called by console to output data</td>
</tr>
<tr>
<td>virtual void Error (const Util::String &amp;s)</td>
<td>called by console with serious error</td>
</tr>
<tr>
<td>virtual void Warning (const Util::String &amp;s)</td>
<td>called by console to output warning</td>
</tr>
<tr>
<td>virtual void DebugOut (const Util::String &amp;s)</td>
<td>called by console to output debug string</td>
</tr>
<tr>
<td>bool IsOpen () const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td>virtual void Update ()</td>
<td>called by Console::Update()</td>
</tr>
<tr>
<td>virtual void Confirm (const Util::String &amp;s)</td>
<td>called by console to display a confirmation message box</td>
</tr>
<tr>
<td>virtual bool HasInput ()</td>
<td>return true if input is available</td>
</tr>
<tr>
<td>virtual Util::String GetInput ()</td>
<td>read available input</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td></td>
</tr>
</tbody>
</table>
```cpp
bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
```
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

int
Core::RefCounted::GetRefCount () const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef () [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release () [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName () const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC () const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks () [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::MediaType
IO::MediaType Class Reference

#include <mediatype.h>
Detailed Description

Encapsulates a MIME conformant media type description (text/plain, image/jpg, etc...).

(C) 2006 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MediaType ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>MediaType (const Util::String &amp;str)</strong></td>
<td>init constructor from string</td>
</tr>
<tr>
<td><strong>MediaType (const Util::String &amp;type, const Util::String &amp;subType)</strong></td>
<td>init constructor from type and subtype</td>
</tr>
<tr>
<td><strong>MediaType (const MediaType &amp;rhs)</strong></td>
<td>copy constructor</td>
</tr>
<tr>
<td><strong>void operator= (const MediaType &amp;rhs)</strong></td>
<td>assignment operator</td>
</tr>
<tr>
<td><strong>bool operator== (const MediaType &amp;rhs)</strong></td>
<td>equality operator</td>
</tr>
<tr>
<td><strong>bool operator!= (const MediaType &amp;rhs)</strong></td>
<td>inequality operator</td>
</tr>
<tr>
<td><strong>bool IsValid () const</strong></td>
<td>return true if not empty</td>
</tr>
<tr>
<td><strong>void Clear ()</strong></td>
<td>clear the media type object</td>
</tr>
<tr>
<td><strong>void Set (const Util::String &amp;str)</strong></td>
<td>set as string (must be of the form &quot;xxx/yyy&quot;)</td>
</tr>
<tr>
<td><strong>void Set (const Util::String &amp;type, const Util::String &amp;subType)</strong></td>
<td>set as type and subtype</td>
</tr>
<tr>
<td><strong>Util::String AsString () const</strong></td>
<td>return as string</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetType () const</strong></td>
<td>get type</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetSubType () const</strong></td>
<td>get subtype</td>
</tr>
</tbody>
</table>
IO::MemoryStream
# include <memorystream.h>

Inheritance diagram for IO::MemoryStream:
Detailed Description

Implements a stream class which writes to and reads from system RAM. Memory streams provide memory mapping for fast direct read/write access.

(C) 2006 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enum</code> AccessMode</td>
<td>access modes</td>
</tr>
<tr>
<td><code>enum</code> AccessPattern</td>
<td>access preferred pattern</td>
</tr>
<tr>
<td><code>enum</code> SeekOrigin</td>
<td>seek origins</td>
</tr>
<tr>
<td><code>typedef int</code> Position</td>
<td>typedefs</td>
</tr>
</tbody>
</table>
Public Member Functions

**MemoryStream ()**
constructor

virtual **~MemoryStream ()**
destructor

virtual bool **CanRead () const**
memory streams support reading

virtual bool **CanWrite () const**
memory streams support writing

virtual bool **CanSeek () const**
memory streams support seeking

virtual bool **CanBeMapped () const**
memory streams are mappable

virtual void **SetSize** (Size s)
set new size of the stream in bytes

virtual Size **GetSize () const**
get the size of the stream in bytes

virtual Position **GetPosition () const**
get the current position of the read/write cursor

virtual bool **Open ()**
open the stream

virtual void **Close ()**
close the stream

virtual void **Write** (const void *ptr, Size numBytes)
directly write to the stream

virtual Size **Read** (void *ptr, Size numBytes)
directly read from the stream

virtual void **Seek** (Offset offset, SeekOrigin origin)
seek in stream

virtual bool **Eof () const**
return true if end-of-stream reached

virtual void * **Map ()**
map for direct memory-access

virtual void **Unmap ()**
unmap a mapped stream

void * **GetRawPointer** () const
get a direct "raw" pointer to the data

void **SetURI** (const **URI** &u)
set stream location as **URI**

const **URI** & **GetURI** () const
get stream **URI**

virtual void **SetSize** (Size s)
set a new size for the stream

void **SetAccessMode** (**AccessMode** m)
set the access mode of the stream (default is ReadAccess)

**AccessMode** **GetAccessMode** () const
get the access mode of the stream

void **SetAccessPattern** (**AccessPattern** p)
set the preferred access pattern (default is Sequential)

**AccessPattern** **GetAccessPattern** () const
get the preferred access pattern

void **SetMediaType** (const **MediaType** &t)
set optional media type of stream content

const **MediaType** & **GetMediaType** () const
get optional media type

bool **IsOpen** () const
return true if currently open

virtual void **Write** (const void *ptr, Size numBytes)
directly write to the stream

virtual Size **Read** (void *ptr, Size numBytes)
directly read from the stream

virtual void **Seek** (Offset offset, **SeekOrigin** origin)
seek in stream

virtual void **Flush** ()
flush unsaved data

bool **IsMapped** () const
return true if stream is currently mapped to memory

int **GetRefCount** () const
get the current refcount

void **AddRef** ()
increment refcount by one
**void** Release ()

*decrement refcount and destroy object if refcount is zero*

**bool** IsInstanceOf (const Rtti &rtti) const

*return true if this object is instance of given class*

**bool** IsInstanceOf (const Util::String &className) const

*return true if this object is instance of given class by string*

**bool** IsInstanceOf (const Util::FourCC &classFourCC) const

*return true if this object is instance of given class by fourcc*

**bool** IsA (const Rtti &rtti) const

*return true if this object is instance of given class, or a derived class*

**bool** IsA (const Util::String &rttiName) const

*return true if this object is instance of given class, or a derived class, by string*

**bool** IsA (const Util::FourCC &rttiFourCC) const

*return true if this object is instance of given class, or a derived class, by fourcc*

**const Util::String &** GetClassName () const

*get the class name*

**Util::FourCC** GetClassFourCC () const

*get the class FourCC code*
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

`bool IO::MemoryStream::Open()` [virtual]

open the stream

Open the stream for reading or writing. The stream may already contain data if it has been opened/closed before.

Reimplemented from `IO::Stream`.

`void IO::MemoryStream::Close()` [virtual]

close the stream

Close the stream. The contents of the stream will remain intact until destruction of the object, so that the same data may be accessed or modified during a later session.

Reimplemented from `IO::Stream`.

`void * IO::MemoryStream::Map()` [virtual]

map for direct memory-access

Map the stream for direct memory access. This is much faster then reading/writing, but less flexible. A mapped stream cannot grow, instead the allowed memory range is determined by `GetSize()`. The read/writer must take special care to not read or write past the memory buffer boundaries!

Reimplemented from `IO::Stream`.

`void IO::MemoryStream::Unmap()` [virtual]

unmap a mapped stream
Unmap a memory-mapped stream.

Reimplemented from `IO::Stream`.

```cpp
void *
IO::MemoryStream::GetRawPointer() const
```

get a direct "raw" pointer to the data

Get a direct pointer to the raw data. This is a convenience method and only works for memory streams. NOTE: writing new data to the stream may/will result in an invalid pointer, don't keep the returned pointer around between writes!

```cpp
void
IO::Stream::SetURI(const URI & u) [inherited]
```

set stream location as `URI`

Set the `URI` of the stream as string. The `URI` identifies the source resource of the stream.

```cpp
const URI &
IO::Stream::GetURI() const [inherited]
```

get stream `URI`

Get the `URI` of the stream as string.

```cpp
void
IO::Stream::SetSize(Size s) [virtual, inherited]
```

set a new size for the stream

This sets a new size for the stream. Not all streams support this method. If the new size if smaller then the existing size, the contents will be clipped.

```cpp
void
IO::Stream::SetAccessMode(AccessMode m) [inherited]
```
set the access mode of the stream (default is ReadAccess)

This method sets the intended access mode of the stream. The actual behaviour depends on the implementation of the derived class. The default is ReadWrite.

```
Stream::AccessMode
IO::Stream::GetAccessMode( ) const [inherited]
```

get the access mode of the stream

Get the access mode of the stream.

```
void
IO::Stream::SetAccessPattern( AccessPattern p ) [inherited]
```

set the prefered access pattern (default is Sequential)

Set the prefered access pattern of the stream. This can be Random or Sequential. This is an optional flag to improve performance with some stream implementations. The default is sequential. The pattern cannot be changed while the stream is open.

```
Stream::AccessPattern
IO::Stream::GetAccessPattern( ) const [inherited]
```

get the prefered access pattern

Get the currently set prefered access pattern of the stream.

```
bool
IO::Stream::IsOpen( ) const [inherited]
```

return true if currently open

Return true if the stream is currently open.

```
void
IO::Stream::Write( void * ptr,
const Size numBytes ) const [virtual, inherited]
```
directly write to the stream

Write raw data to the stream. For more convenient writing, attach the stream to an **IO::StreamWriter** object. This method is only valid if the stream class returns true in **CanWrite()**.

```
Stream::Size
IO::Stream::Read (*ptr,
    Size numBytes)
```

[virtual, inherited]

directly read from the stream

Read raw data from the stream. For more convenient reading, attach the stream to an **IO::StreamReader** object. The method returns the number of bytes actually read. This method is only valid if the stream class returns true in **CanRead()**. Returns the number of bytes actually read from the stream, this may be less than numBytes, or 0 if end-of-stream is reached.

```
void
IO::Stream::Seek (Offset offset,
    SeekOrigin origin)
```

[virtual, inherited]

seek in stream

Move the read/write cursor to a new position, returns the new position in the stream. This method is only supported if the stream class returns true in **CanSeek()**.

```
void
IO::Stream::Flush ( )
```

[virtual, inherited]

flush unsaved data

Flush any unsaved data. Note that unsaved data will also be flushed automatically when the stream is closed.

Reimplemented in **IO::FileStream**.
bool ( ) const [inherited]
IO::Stream::IsMapped

return true if stream is currently mapped to memory

Returns true if the stream is currently mapped.

int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void ( )
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
IO::SchemeRegistry
IO::SchemeRegistry Class Reference

#include <schemeregistry.h>

Inheritance diagram for IO::SchemeRegistry:

```
Core::RefCounted
|-- IO::SchemeRegistry
```
Detailed Description

Central registry for **URI** schemes, associates an **URI** scheme (e.g. http, file, ...) with a Nebula3 stream class.

(C) 2008 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SchemeRegistry ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~SchemeRegistry ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual <strong>Setup ()</strong></td>
<td>setup the scheme registry (may only be called at app startup by the main thread)</td>
</tr>
<tr>
<td><strong>Discard ()</strong></td>
<td>discard the scheme registry</td>
</tr>
<tr>
<td><strong>IsValid () const</strong></td>
<td>return true if the scheme registry is valid</td>
</tr>
<tr>
<td><strong>RegisterUriScheme (const Util::String &amp;uriScheme, const Core::Rtti &amp;classRtti)</strong></td>
<td>associate an uri scheme with a stream class</td>
</tr>
<tr>
<td><strong>UnregisterUriScheme (const Util::String &amp;uriScheme)</strong></td>
<td>unregister an uri scheme</td>
</tr>
<tr>
<td><strong>IsUriSchemeRegistered (const Util::String &amp;uriScheme) const</strong></td>
<td>return true if an uri scheme has been registered</td>
</tr>
<tr>
<td><strong>GetStreamClassByUriScheme (const Util::String &amp;uriScheme) const</strong></td>
<td>get the registered stream class for an uri scheme</td>
</tr>
<tr>
<td><strong>GetAllRegisteredUriSchemes () const</strong></td>
<td>get an array of all registered schemes</td>
</tr>
<tr>
<td><strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

```cpp
void IO::SchemeRegistry::RegisterUriScheme(
    const Util::String& uriScheme,
    const Core::Rtti& classRtti
)
```

associate an uri scheme with a stream class

Associates an URI scheme with a stream class. If the same URI scheme is already registered, the old association will be overwritten.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String& Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name
Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::Stream
#include <stream.h>

Inheritance diagram for IO::Stream:

```
Core::RefCounted
  ↓
IO::Stream
  ↓
IO::FileStream  IO::MemoryStream  IO::ZipFileStream
```
Detailed Description

Offers an abstract interface for read/write access to a sequence of bytes. **Base** class for all classes which need to provide a read/write interface to data, like a **FileStream**, a **MemoryStream**, etc...

**Stream** objects can be accessed directly, or through stream reader and writer classes, which offer specific read/writing interfaces to streams.

(C) 2006 Radon Labs GmbH
# Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><strong>AccessMode</strong></td>
</tr>
<tr>
<td></td>
<td>access modes</td>
</tr>
<tr>
<td>enum</td>
<td><strong>AccessPattern</strong></td>
</tr>
<tr>
<td></td>
<td>access preferred pattern</td>
</tr>
<tr>
<td>enum</td>
<td><strong>SeekOrigin</strong></td>
</tr>
<tr>
<td></td>
<td>seek origins</td>
</tr>
<tr>
<td>typedef int</td>
<td><strong>Position</strong></td>
</tr>
<tr>
<td></td>
<td>typedefs</td>
</tr>
</tbody>
</table>
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream()</td>
<td>constructor</td>
</tr>
<tr>
<td>~Stream()</td>
<td>destructor</td>
</tr>
<tr>
<td>void SetURI(const URI &amp;u)</td>
<td>set stream location as URI</td>
</tr>
<tr>
<td>const URI &amp; GetURI() const</td>
<td>get stream URI</td>
</tr>
<tr>
<td>virtual bool CanRead() const</td>
<td>return true if the stream supports reading</td>
</tr>
<tr>
<td>virtual bool CanWrite() const</td>
<td>return true if the stream supports writing</td>
</tr>
<tr>
<td>virtual bool CanSeek() const</td>
<td>return true if the stream supports seeking</td>
</tr>
<tr>
<td>virtual bool CanBeMapped() const</td>
<td>return true if the stream provides direct memory access</td>
</tr>
<tr>
<td>virtual void SetSize(Size s)</td>
<td>set a new size for the stream</td>
</tr>
<tr>
<td>virtual Size GetSize() const</td>
<td>get the size of the stream in bytes</td>
</tr>
<tr>
<td>virtual Position GetPosition() const</td>
<td>get the current position of the read/write cursor</td>
</tr>
<tr>
<td>void SetAccessMode(AccessMode m)</td>
<td>set the access mode of the stream (default is ReadAccess)</td>
</tr>
<tr>
<td>AccessMode GetAccessMode() const</td>
<td>get the access mode of the stream</td>
</tr>
<tr>
<td>void SetAccessPattern(AccessPattern p)</td>
<td>set the prefered access pattern (default is Sequential)</td>
</tr>
<tr>
<td>AccessPattern GetAccessPattern() const</td>
<td>get the prefered access pattern</td>
</tr>
<tr>
<td>void SetMediaType(const MediaType &amp;t)</td>
<td>set optional media type of stream content</td>
</tr>
<tr>
<td>const MediaType &amp; GetMediaType() const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td><code>get optional media type</code></td>
<td></td>
</tr>
<tr>
<td><code>virtual bool Open()</code></td>
<td>open the stream</td>
</tr>
<tr>
<td><code>virtual void Close()</code></td>
<td>close the stream</td>
</tr>
<tr>
<td><code>bool IsOpen () const</code></td>
<td>return true if currently open</td>
</tr>
<tr>
<td><code>virtual void Write (const void *ptr, Size numBytes)</code></td>
<td>directly write to the stream</td>
</tr>
<tr>
<td><code>virtual Size Read (void *ptr, Size numBytes)</code></td>
<td>directly read from the stream</td>
</tr>
<tr>
<td><code>virtual void Seek (Offset offset, SeekOrigin origin)</code></td>
<td>seek in stream</td>
</tr>
<tr>
<td><code>virtual void Flush ()</code></td>
<td>flush unsaved data</td>
</tr>
<tr>
<td><code>virtual bool Eof () const</code></td>
<td>return true if end-of-stream reached</td>
</tr>
<tr>
<td><code>virtual void Map ()</code></td>
<td>map stream to memory</td>
</tr>
<tr>
<td><code>virtual void Unmap ()</code></td>
<td>unmap stream</td>
</tr>
<tr>
<td><code>bool IsMapped () const</code></td>
<td>return true if stream is currently mapped to memory</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class</em></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by string</em></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by fourcc</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>Util::String</strong> &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
set stream location as **URI**

Set the **URI** of the stream as string. The **URI** identifies the source resource of the stream.

get stream **URI**

Get the **URI** of the stream as string.

return true if the stream supports reading

This method must return true if the derived stream class supports reading.

Reimplemented in **IO::FileStream**, **IO::MemoryStream**, and **IO::ZipFileStream**.

return true if the stream supports writing

This method must return true if the derived stream class supports writing.

Reimplemented in **IO::FileStream**, **IO::MemoryStream**, and **IO::ZipFileStream**.
bool IO::Stream::CanSeek ( ) const [virtual]
return true if the stream supports seeking
This method must return true if the derived stream supports seeking.
Reimplemented in IO::FileStream, IO::MemoryStream, and IO::ZipFileStream.

bool IO::Stream::CanBeMapped ( ) const [virtual]
return true if the stream provides direct memory access
This method must return true if the stream supports direct memory access through the Map()/Unmap() methods.
Reimplemented in IO::FileStream, IO::MemoryStream, and IO::ZipFileStream.

void IO::Stream::SetSize (Size s ) [virtual]
set a new size for the stream
This sets a new size for the stream. Not all streams support this method. If the new size if smaller then the existing size, the contents will be clipped.

Stream::Size IO::Stream::GetSize ( ) const [virtual]
get the size of the stream in bytes
This method returns the size of the stream in bytes.
Reimplementeded in IO::FileStream, IO::MemoryStream, and IO::ZipFileStream.

Stream::Position IO::Stream::GetPosition ( ) const [virtual]
get the current position of the read/write cursor

This method returns the current position of the read/write cursor.

Reimplemented in **IO::FileStream**, **IO::MemoryStream**, and **IO::ZipFileStream**.

```cpp
void IO::Stream::SetAccessMode(AccessMode m)
```

set the access mode of the stream (default is ReadAccess)

This method sets the intended access mode of the stream. The actual behaviour depends on the implementation of the derived class. The default is ReadWrite.

```cpp
Stream::AccessMode IO::Stream::GetAccessMode() const
```

get the access mode of the stream

Get the access mode of the stream.

```cpp
void IO::Stream::SetAccessPattern(AccessPattern p)
```

set the preferred access pattern (default is Sequential)

Set the preferred access pattern of the stream. This can be Random or Sequential. This is an optional flag to improve performance with some stream implementations. The default is sequential. The pattern cannot be changed while the stream is open.

```cpp
Stream::AccessPattern IO::Stream::GetAccessPattern() const
```

get the preferred access pattern

Get the currently set preferred access pattern of the stream.

```cpp
bool IO::Stream::Open() [virtual]
```
open the stream

Open the stream. Only one thread may open a stream at any time. Returns true if succeeded.

Reimplemented in IO::FileStream, IO::MemoryStream, and IO::ZipFileStream.

```cpp
void IO::Stream::Close()
```

close the stream

Closes the stream.

Reimplemented in IO::FileStream, IO::MemoryStream, and IO::ZipFileStream.

```cpp
bool IO::Stream::IsOpen()
```

return true if currently open

Return true if the stream is currently open.

```cpp
void IO::Stream::Write(const * ptr,
                       const Size numBytes)
```

directly write to the stream

Write raw data to the stream. For more convenient writing, attach the stream to an IO::StreamWriter object. This method is only valid if the stream class returns true in CanWrite().

```cpp
void IO::Stream::Read(* ptr,
                      Size numBytes)
```
directly read from the stream

Read raw data from the stream. For more convenient reading, attach the stream to an **IO::StreamReader** object. The method returns the number of bytes actually read. This method is only valid if the stream class returns true in **CanRead()**. Returns the number of bytes actually read from the stream, this may be less than numBytes, or 0 if end-of-stream is reached.

```cpp
void IO::Stream::Seek( Offset offset,
                      SeekOrigin origin         [virtual]
)
```

seek in stream

Move the read/write cursor to a new position, returns the new position in the stream. This method is only supported if the stream class returns true in **CanSeek()**.

```cpp
void IO::Stream::Flush( ) [virtual]
```

flush unsaved data

Flush any unsaved data. Note that unsaved data will also be flushed automatically when the stream is closed.

Reimplemented in **IO::FileStream**.

```cpp
bool IO::Stream::Eof( ) const [virtual]
```

return true if end-of-stream reached

Return true if the read/write cursor is at the end of the stream.

Reimplemented in **IO::FileStream**, **IO::MemoryStream**, and **IO::ZipFileStream**.

```cpp
void * IO::Stream::Map( ) [virtual]
```
map stream to memory

If the stream provides memory mapping, this method will return a pointer to the beginning of the stream data in memory. The application is free to read and write to the stream through direct memory access. Special care must be taken to not read or write past the end of the mapped data (indicated by `GetSize()`). The normal `Read()`/`Write()` method are not valid while the stream is mapped, also the read/write cursor position will not be updated.

Reimplemented in `IO::FileStream`, `IO::MemoryStream`, and `IO::ZipFileStream`.

```cpp
void IO::Stream::Unmap() [virtual]
```

unmap stream

This will unmap a memory-mapped stream.

Reimplemented in `IO::FileStream`, `IO::MemoryStream`, and `IO::ZipFileStream`.

```cpp
bool IO::Stream::IsMapped() const
```

return true if stream is currently mapped to memory

Returns true if the stream is currently mapped.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

---

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:46 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::StreamReader
#include <streamreader.h>

Inheritance diagram for IO::StreamReader:
Detailed Description

Stream reader classes provide a specialized read-interface for a stream. This is the abstract base class for all stream readers. It is possible to attach any number of readers and writers to the same stream.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StreamReader ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~StreamReader ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void SetStream (const Ptr&lt; Stream &gt;&amp; s)</td>
<td>set stream to read from</td>
</tr>
<tr>
<td>const Ptr&lt; Stream &gt;&amp; &amp; GetStream () const</td>
<td>get currently set stream</td>
</tr>
<tr>
<td>bool HasStream () const</td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td>bool Eof () const</td>
<td>return true if the stream has reached EOF</td>
</tr>
<tr>
<td>virtual bool Open ()</td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td>end reading from the stream</td>
</tr>
<tr>
<td>bool IsOpen () const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class by rtti</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
<td></td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
<td></td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetClassName</strong> () const</td>
<td></td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td><strong>Util::FourCC</strong> <strong>GetClassFourCC</strong> () const</td>
<td></td>
</tr>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Member Function Documentation

**void IO::StreamReader::SetStream**  
const **Ptr< Stream > & s**  

set stream to read from

Attaches the reader to a stream. This will increment the refcount of the stream.

Reimplemented in **Messaging::MessageReader**.

**const Ptr< Stream > &**  
**IO::StreamReader::GetStream**  
const

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use **HasStream()** to determine if a stream is attached.

**bool IO::StreamReader::HasStream**  
const

return true if a stream is set

Returns true if a stream is attached to the reader.

**int Core::RefCounted::GetRefCount**  
const [inline, inherited]

get the current refcount

Return the current refcount of the object.

**void Core::RefCounted::AddRef**  
[inline, inherited]

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::StreamWriter
#include <streamwriter.h>

Inheritance diagram for IO::StreamWriter:

```
Core::RefCounted

IO::StreamWriter

IO::BinaryWriter

IO::TextWriter

IO::XmlWriter

Messaging::MessageWriter

Http::HtmlPageWriter

Http::HttpRequestWriter

Http::HttpResponseWriter

Http::SvgPageWriter
```
Detailed Description

Stream writer classes provide a specialized write-interface for a stream. This is the abstract base class for all stream writers. It is possible to attach any number of readers and writers to the same stream.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StreamWriter ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~StreamWriter ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>void SetStream (const Ptr&lt; Stream&gt; &amp;s)</td>
<td>Set stream to write to</td>
</tr>
<tr>
<td>const Ptr&lt; Stream&gt; &amp;</td>
<td>GetStream () const</td>
</tr>
<tr>
<td>void HasStream () const</td>
<td>Return true if a stream is set</td>
</tr>
<tr>
<td>virtual bool Open ()</td>
<td>Begin reading from the stream</td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td>End reading from the stream</td>
</tr>
<tr>
<td>bool IsOpen () const</td>
<td>Return true if currently open</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>
bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
Static Public Member Functions

```c
static void DumpRefCountingLeaks ()
    dump refcounting leaks, call at end of application (NEBUA3_DEBUG builds only!)
```
Member Function Documentation

```cpp
void IO::StreamWriter::SetStream(const Ptr<Stream> & s)
```

set stream to write to

Attaches the writer to a stream. This will increment the refcount of the stream.

Reimplemented in `Messaging::MessageWriter`.

```cpp
const Ptr<Stream> & IO::StreamWriter::GetStream() const
```

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use `HasStream()` to determine if a stream is attached.

```cpp
bool IO::StreamWriter::HasStream() const
```

return true if a stream is set

Returns true if a stream is attached to the writer.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
IO::TextReader
# include <textreader.h>

Inheritance diagram for IO::TextReader:
Detailed Description

A friendly interface for reading text data from a stream.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextReader()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~TextReader()</td>
<td>Destructor</td>
</tr>
<tr>
<td>unsigned char ReadChar()</td>
<td>read a single character from the stream</td>
</tr>
<tr>
<td>Util::String ReadLine()</td>
<td>read until next newline</td>
</tr>
<tr>
<td>Util::String ReadAll()</td>
<td>read entire stream into a string object</td>
</tr>
<tr>
<td>Util::Array<a href="">Util::String</a> ReadAllLines()</td>
<td>read entire stream as lines into string array</td>
</tr>
<tr>
<td>void SetStream(const Ptr&lt;Stream&gt; &amp;s)</td>
<td>set stream to read from</td>
</tr>
<tr>
<td>const Ptr&lt;Stream&gt; &amp; GetStream() const</td>
<td>get currently set stream</td>
</tr>
<tr>
<td>bool HasStream() const</td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td>bool Eof() const</td>
<td>return true if the stream has reached EOF</td>
</tr>
<tr>
<td>virtual bool Open()</td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td>virtual void Close()</td>
<td>end reading from the stream</td>
</tr>
<tr>
<td>bool IsOpen() const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>check if the object is an instance of the specified Rtti</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>static void</td>
<td><code>DumpRefCountingLeaks()</code></td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void IO::StreamReader::SetStream(const Ptr<Stream> & s) [inherited]
```

description:
set stream to read from

Attaches the reader to a stream. This will increment the refcount of the stream.

Reimplemented in Messaging::MessageReader.

```cpp
const Ptr<Stream> & IO::StreamReader::GetStream() const [inherited]
```

description:
get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use HasStream() to determine if a stream is attached.

```cpp
bool IO::StreamReader::HasStream() const [inherited]
```

description:
return true if a stream is set

Returns true if a stream is attached to the reader.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

description:
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

description:
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**IO::TextWriter**
IO::TextWriter Class Reference

#include <textwriter.h>

Inheritance diagram for IO::TextWriter:

```
Core::RefCounted
    ↑
IO::StreamWriter
    ↑
IO::TextWriter
```
Detailed Description

A friendly interface for writing text data to a stream.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TextWriter()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~TextWriter()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>void <code>WriteChar</code> (unsigned char c)</td>
<td>write a single character</td>
</tr>
<tr>
<td>void <code>WriteString</code> (const Util::String &amp;str)</td>
<td>write a string</td>
</tr>
<tr>
<td>void <code>WriteFormatted</code> (const char *fmtString,...)</td>
<td>write some formatted text</td>
</tr>
<tr>
<td>void <code>WriteLine</code> (const Util::String &amp;line)</td>
<td>write a line of text and append a new-line</td>
</tr>
<tr>
<td>void <code>WriteLines</code> (const Util::Array&lt; Util::String &gt; &amp;lines)</td>
<td>write a number of lines, separated by new-lines</td>
</tr>
<tr>
<td>template&lt;typename T&gt;</td>
<td>void <code>Write</code> (const T &amp;t)</td>
</tr>
<tr>
<td>void <code>SetStream</code> (const Ptr&lt; Stream &gt; &amp;s)</td>
<td>set stream to write to</td>
</tr>
<tr>
<td>const Ptr&lt; Stream &gt; &amp;</td>
<td><code>GetStream()</code> const</td>
</tr>
<tr>
<td>bool <code>HasStream</code> () const</td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td>virtual bool <code>Open</code> ()</td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td>virtual void <code>Close</code> ()</td>
<td>end reading from the stream</td>
</tr>
<tr>
<td>bool <code>isOpen</code> () const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td>int <code>GetRefCount</code> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef</code> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName() const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC() const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
void IO::StreamWriter::SetStream(const Ptr<Stream> & s) [inherited]

set stream to write to

Attaches the writer to a stream. This will increment the refcount of the stream.

Reimplemented in Messaging::MessageWriter.

constPtr<Stream> & IO::StreamWriter::GetStream ( ) const [inherited]

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use HasStream() to determine if a stream is attached.

bool IO::StreamWriter::HasStream ( ) const [inherited]

return true if a stream is set

Returns true if a stream is attached to the writer.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump ref counting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::URI
IO::URI Class Reference

#include <uri.h>
Detailed Description

An **URI** object can split a Uniform Resource Identifier string into its components or build a string from **URI** components. Please note that the memory footprint of an **URI** object is always bigger than a pure String object, so if memory usage is of concern, it is advised to keep paths as String objects around, and only use **URI** objects to encode and decode them.

An **URI** is made of the following components, where most of them are optional:

```
Scheme://UserInfo:Port/LocalPath Fragment?Query
```

Example URIs:

- http://user:password@www.myserver.com:8080/index.html#main
- file:///c:/temp/bla.txt
- file://SambaServer/temp/blub.txt

Note that assigns will be resolved before splitting a **URI** into its components, for instance the assign "textures" could be defined as:

```
Assign("textures", "http://www.dataserv.com/myapp/textures/");
```

So a path to a texture **URI** could be defined as:

```
URI("textures:mytex.dds")
```

Which would actually resolve into:

```
http://www.dataserv.com/myapp/textures/mytex.dds
```

Decoding into components happens in the init constructor or the **Set()** method in the following steps:

- resolve any assigns in the original string
- split into Scheme, Host and Path blocks
- resolve Host and Path blocks further

Encoding from components into string happens in the `AsString()` method in the following steps:

- concatenate **URI** string from components
- convert part of the string back into an existing assign

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>URI()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>URI(const Util::String &amp;s)</code></td>
<td>init constructor</td>
</tr>
<tr>
<td><code>URI(const char *s)</code></td>
<td>init constructor</td>
</tr>
<tr>
<td><code>URI(const URI &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>void operator=(const URI &amp;rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td><code>bool operator==(const URI &amp;rhs)</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const URI &amp;rhs)</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>void Set(const Util::String &amp;s)</code></td>
<td>set complete URI string</td>
</tr>
<tr>
<td><code>Util::String AsString()</code></td>
<td>return as concatenated string</td>
</tr>
<tr>
<td><code>bool IsEmpty()</code></td>
<td>return true if the URI is empty</td>
</tr>
<tr>
<td><code>bool IsValid()</code></td>
<td>return true if the URI is not empty</td>
</tr>
<tr>
<td><code>void Clear()</code></td>
<td>clear the URI</td>
</tr>
<tr>
<td><code>void SetScheme(const Util::String &amp;s)</code></td>
<td>set Scheme component (ftp, http, etc...)</td>
</tr>
<tr>
<td><code>const Util::String &amp; Scheme()</code></td>
<td>get Scheme component (default is file)</td>
</tr>
<tr>
<td><code>void SetUserInfo(const Util::String &amp;s)</code></td>
<td>set UserInfo component</td>
</tr>
<tr>
<td><code>const Util::String &amp; UserInfo()</code></td>
<td>get UserInfo component (can be empty)</td>
</tr>
<tr>
<td><code>void SetHost(const Util::String &amp;s)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const Util::String &amp; Host () const</code></td>
<td>set Host component</td>
</tr>
<tr>
<td><code>get Host component (can be empty)</code></td>
<td></td>
</tr>
<tr>
<td><code>void SetPort (const Util::String &amp;s)</code></td>
<td>set Port component</td>
</tr>
<tr>
<td><code>const Util::String &amp; Port () const</code></td>
<td>get Port component (can be empty)</td>
</tr>
<tr>
<td><code>void SetLocalPath (const Util::String &amp;s)</code></td>
<td>set LocalPath component</td>
</tr>
<tr>
<td><code>const Util::String &amp; LocalPath () const</code></td>
<td>get LocalPath component</td>
</tr>
<tr>
<td><code>void AppendLocalPath (const Util::String &amp;pathComponent)</code></td>
<td>append an element to the local path component</td>
</tr>
<tr>
<td><code>void SetFragment (const Util::String &amp;s)</code></td>
<td>set Fragment component</td>
</tr>
<tr>
<td><code>const Util::String &amp; Fragment () const</code></td>
<td>get Fragment component</td>
</tr>
<tr>
<td><code>void SetQuery (const Util::String &amp;s)</code></td>
<td>set Query component</td>
</tr>
<tr>
<td><code>const Util::String &amp; Query () const</code></td>
<td>get Query component (can be empty)</td>
</tr>
<tr>
<td><code>Util::Dictionary&lt; Util::String, Util::String &gt; ParseQuery () const</code></td>
<td>parse query parameters into a dictionary</td>
</tr>
<tr>
<td><code>Util::String GetTail () const</code></td>
<td>get the &quot;tail&quot; (path, query and fragment)</td>
</tr>
<tr>
<td><code>Util::String GetHostAndLocalPath () const</code></td>
<td>get the host and path without scheme</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void IO::URI::AppendLocalPath(const Util::String& pathComponent)
```

append an element to the local path component

Appends an element to the local path. Automatically inserts a path delimiter "/".

```cpp
Dictionary< String, String > const IO::URI::ParseQuery()
```

parse query parameters into a dictionary

This parses the query part of the URI (in the form `param1=value&param2=value&param3=value ...`) into a dictionary. Ill-formatted query fragments will be ignored.

```cpp
String const IO::URI::GetTail()
```

get the "tail" (path, query and fragment)

This returns the "tail", which is the local path, the fragment and the query concatenated into one string.

```cpp
String const IO::URI::GetHostAndLocalPath()
```

get the host and path without scheme

Returns the host and local path in the form "/host/localpath". If no host has been set, only "/localpath" will be returned.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::XmlReader
IO::XmlReader Class Reference

#include <xmlreader.h>

Inheritance diagram for IO::XmlReader:

```
Core::RefCounted
  ↓
IO::StreamReader
  ↓
IO::XmlReader
```
Detailed Description

Reads XML formatted data with random access from a stream using TinyXML as backend. The XML document is represented as a tree of nodes, which can be navigated and queried.

(C) 2006 Radon Labs GmbH
### Public Member Functions

**XmlReader ()**

*constructor*

**virtual ~XmlReader ()**

*destructor*

**virtual bool Open ()**

*begin reading from the stream*

**virtual void Close ()**

*end reading from the stream*

**bool HasNode (const Util::String &path) const**

*return true if node exists*

**Util::String GetCurrentNodeName () const**

*get short name of current node*

**Util::String GetCurrentNodePath () const**

*get path to current node*

**int GetCurrentNodeLineNumber () const**

*returns the line number of the current node*

**void SetToNode (const Util::String &path)**

*set current node as path*

**bool SetToFirstChild (const Util::String &name="")**

*set current node to first child node, return false if no child exists*

**bool SetToNextChild (const Util::String &name="")**

*set current node to next sibling node, return false if no more sibling exists*

**bool SetToParent ()**

*set current node to parent, return false if no parent exists*

**bool HasAttr (const char *attr) const**

*return true if matching attribute exists on current node*

**Util::Array< Util::String > GetAttrs () const**

*return names of all attrs on current node*

**bool HasContent () const**

*return true if current node has embedded content*

**Util::String GetContent () const**

*return embedded content of current node*
<table>
<thead>
<tr>
<th>Datatype</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Util::String</td>
<td><strong>GetString</strong> (const char *attr) const</td>
<td>get string attribute value from current node</td>
</tr>
<tr>
<td>bool</td>
<td><strong>GetBool</strong> (const char *attr) const</td>
<td>get bool attribute value from current node</td>
</tr>
<tr>
<td>int</td>
<td><strong>GetInt</strong> (const char *attr) const</td>
<td>get int attribute value from current node</td>
</tr>
<tr>
<td>float</td>
<td><strong>GetFloat</strong> (const char *attr) const</td>
<td>get float attribute value from current node</td>
</tr>
<tr>
<td>Math::float2</td>
<td><strong>GetFloat2</strong> (const char *attr) const</td>
<td>get float2 attribute value from current node</td>
</tr>
<tr>
<td>Math::float4</td>
<td><strong>GetFloat4</strong> (const char *attr) const</td>
<td>get float4 attribute value from current node</td>
</tr>
<tr>
<td>Math::matrix44</td>
<td><strong>GetMatrix44</strong> (const char *attr) const</td>
<td>get matrix44 attribute value from current node</td>
</tr>
<tr>
<td>template&lt;typename T&gt;</td>
<td><strong>Get</strong> (const char *attr) const</td>
<td>generic getter for extension types</td>
</tr>
<tr>
<td>Util::String</td>
<td><strong>GetOptString</strong> (const char *attr, const Util::String &amp;defaultValue) const</td>
<td>get optional string attribute value from current node</td>
</tr>
<tr>
<td>bool</td>
<td><strong>GetOptBool</strong> (const char *attr, bool defaultValue) const</td>
<td>get optional bool attribute value from current node</td>
</tr>
<tr>
<td>int</td>
<td><strong>GetOptInt</strong> (const char *attr, int defaultValue) const</td>
<td>get optional int attribute value from current node</td>
</tr>
<tr>
<td>float</td>
<td><strong>GetOptFloat</strong> (const char *attr, float defaultValue) const</td>
<td>get optional float attribute value from current node</td>
</tr>
<tr>
<td>Math::float2</td>
<td><strong>GetOptFloat2</strong> (const char *attr, const Math::float2 &amp;defaultValue) const</td>
<td>get float2 attribute value from current node</td>
</tr>
<tr>
<td>Math::float4</td>
<td><strong>GetOptFloat4</strong> (const char *attr, const Math::float4 &amp;defaultValue) const</td>
<td>get optional float4 attribute value from current node</td>
</tr>
<tr>
<td>Math::matrix44</td>
<td><strong>GetOptMatrix44</strong> (const char *attr, const Math::matrix44 &amp;defaultValue) const</td>
<td>get matrix44 attribute value from current node</td>
</tr>
</tbody>
</table>
void **SetStream** (const Ptr< Stream > &s)
set stream to read from

const Ptr< Stream > & **GetStream** () const
get currently set stream

bool **HasStream** () const
return true if a stream is set

bool **Eof** () const
return true if the stream has reached EOF

bool **IsOpen** () const
return true if currently open

int **GetRefCount** () const
get the current refcount

void **AddRef** ()
increment refcount by one

void **Release** ()
decrement refcount and destroy object if refcount is zero

bool **IsInstanceOf** (const Rtti &rtti) const
return true if this object is instance of given class

bool **IsInstanceOf** (const Util::String &className) const
return true if this object is instance of given class by string

bool **IsInstanceOf** (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool **IsA** (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool **IsA** (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool **IsA** (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & **GetClassName** () const
get the class name

Util::FourCC **GetClassFourCC** () const
get the class FourCC code
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool IO::XmlReader::Open() [virtual]

begin reading from the stream

Opens the stream and reads the content of the stream into TinyXML.

Reimplemented from IO::StreamReader.

bool IO::XmlReader::HasNode(const Util::String & path) const

return true if node exists

This method returns true if the node identified by path exists. Path follows the normal filesystem path conventions, "/" is the separator, "." is the parent node, "." is the current node. An absolute path starts with a "/", a relative path doesn't.

String IO::XmlReader::GetCurrentNodeName() const

get short name of current node

Get the short name (without path) of the current node.

String IO::XmlReader::GetCurrentNodePath() const

get path to current node

This returns the full absolute path of the current node. Path components are separated by slashes.

int IO::XmlReader::GetCurrentNodeLineNumber() const

returns the line number of the current node
This method returns the line number of the current node.

```cpp
void IO::XmlReader::SetToNode ( const Util::String path )
```

set current node as path

Set the node pointed to by the path string as current node. The path may be absolute or relative, following the usual filesystem path conventions. Separator is a slash.

```cpp
bool IO::XmlReader::SetToFirstChild ( const Util::String name = "" )
```

set current node to first child node, return false if no child exists

Sets the current node to the first child node. If no child node exists, the current node will remain unchanged and the method will return false. If name is a valid string, only child element matching the name will be returned. If name is empty, all child nodes will be considered.

```cpp
bool IO::XmlReader::SetToNextChild ( const Util::String name = "" )
```

set current node to next sibling node, return false if no more sibling exists

Sets the current node to the next sibling. If no more children exist, the current node will be reset to the parent node and the method will return false. If name is a valid string, only child element matching the name will be returned. If name is empty, all child nodes will be considered.

```cpp
bool IO::XmlReader::SetToParent ( )
```

set current node to parent, return false if no parent exists

Sets the current node to its parent. If no parent exists, the current
node will remain unchanged and the method will return false.

```cpp
bool IO::XmlReader::HasAttr(const char* name) const
return true if matching attribute exists on current node
```

Return true if an attribute of the given name exists on the current node.

```cpp
Array<String> IO::XmlReader::GetAttrs() const
return names of all attrs on current node
```

Return array with names of all attrs on current node

```cpp
String IO::XmlReader::GetString(const char* name) const
get string attribute value from current node
```

Return the provided attribute as string. If the attribute does not exist the method will fail hard (use HasAttr() to check for its existence).

```cpp
bool IO::XmlReader::GetBool(const char* name) const
get bool attribute value from current node
```

Return the provided attribute as a bool. If the attribute does not exist the method will fail hard (use HasAttr() to check for its existence).

```cpp
int IO::XmlReader::GetInt(const char* name) const
get int attribute value from current node
```

Return the provided attribute as int. If the attribute does not exist the
method will fail hard (use `HasAttr()` to check for its existence).

```cpp
float
IO::XmlReader::GetFloat (const char* name) const
```

get float attribute value from current node

Return the provided attribute as float. If the attribute does not exist the method will fail hard (use `HasAttr()` to check for its existence).

```cpp
float2
IO::XmlReader::GetFloat2 (const char* name) const
```

get float2 attribute value from current node

Return the provided attribute as float2. If the attribute does not exist the method will fail hard (use `HasAttr()` to check for its existence).

```cpp
float4
IO::XmlReader::GetFloat4 (const char* name) const
```

get float4 attribute value from current node

Return the provided attribute as float4. If the attribute does not exist the method will fail hard (use `HasAttr()` to check for its existence).

```cpp
matrix44
IO::XmlReader::GetMatrix44 (const char* name) const
```

get matrix44 attribute value from current node

Return the provided attribute as matrix44. If the attribute does not exist the method will fail hard (use `HasAttr()` to check for its existence).

```cpp
String
IO::XmlReader::GetOptString (const char* name, const Util::String defaultValue)
```
get optional string attribute value from current node

Return the provided optional attribute as string. If the attribute doesn't exist, the default value will be returned.

\[
\text{const bool IO::XmlReader::GetOptBool (char * name, bool defaultValue)}
\]

get optional bool attribute value from current node

Return the provided optional attribute as bool. If the attribute doesn't exist, the default value will be returned.

\[
\text{const int IO::XmlReader::GetOptInt (char * name, int defaultValue)}
\]

get optional int attribute value from current node

Return the provided optional attribute as int. If the attribute doesn't exist, the default value will be returned.

\[
\text{const float IO::XmlReader::GetOptFloat (char * name, float defaultValue)}
\]

get optional float attribute value from current node

Return the provided optional attribute as float. If the attribute doesn't exist, the default value will be returned.

\[
\text{float2 IO::XmlReader::GetOptFloat2 (const char * name,}
\]

get optional float2 attribute value from current node

Return the provided optional attribute as float2. If the attribute doesn't exist, the default value will be returned.
get float2 attribute value from current node

Return the provided optional attribute as float2. If the attribute doesn't exist, the default value will be returned.

```cpp
float4 IO::XmlReader::GetOptFloat4 (const char * name,
const Math::float4 defaultValue &)
```

get optional float4 attribute value from current node

Return the provided optional attribute as float4. If the attribute doesn't exist, the default value will be returned.

```cpp
matrix44 IO::XmlReader::GetOptMatrix44 (const char * name,
const Math::matrix44 defaultValue &)
```

get optional matrix44 attribute value from current node

Return the provided optional attribute as matrix44. If the attribute doesn't exist, the default value will be returned.

```cpp
void IO::StreamReader::SetStream (const Ptr<Stream> & s ) [inherited]
```

set stream to read from

Attaches the reader to a stream. This will increment the refcount of the stream.
Reimplemented in **Messaging::MessageReader**.

```cpp
class IO::StreamReader{
  public:
    StreamReader() const [inherited]

    get currently set stream
    
    StreamReader::GetStream() const [inherited]
    Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use **HasStream()** to determine if a stream is attached.

    bool
    StreamReader::HasStream() const [inherited]
    return true if a stream is set

    return true if a stream is attached to the reader.

    int
    Core::RefCounted::GetRefCount() const [inline, inherited]
    get the current refcount

    Return the current refcount of the object.

    void
    Core::RefCounted::AddRef() [inline, inherited]
    increment refcount by one

    Increment the refcount of the object.

    void
    Core::RefCounted::Release() [inline, inherited]
    decrement refcount and destroy object if refcount is zero

    Declared in **Core**.

    const
    Core::RefCounted::GetClassName() const [inline, inherited]
    get class name

    Util::String
    Core::RefCounted::GetClassName() const [inline, inherited]
    get class name

    const
    Messaging::MessageReader::GetStream() const [inherited]
    get currently set stream

    StreamReader::GetStream() const [inherited]
    Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use **HasStream()** to determine if a stream is attached.

    bool
    StreamReader::HasStream() const [inherited]
    return true if a stream is set

    return true if a stream is attached to the reader.

    int
    Core::RefCounted::GetRefCount() const [inline, inherited]
    get the current refcount

    Return the current refcount of the object.

    void
    Core::RefCounted::AddRef() [inline, inherited]
    increment refcount by one

    Increment the refcount of the object.

    void
    Core::RefCounted::Release() [inline, inherited]
    decrement refcount and destroy object if refcount is zero

    Declared in **Core**.

    const
    Core::RefCounted::GetClassName() const [inline, inherited]
    get class name

    Util::String
    Core::RefCounted::GetClassName() const [inline, inherited]
    get class name
```
get the class name

Get the class name of the object.

`Util::FourCC`  
Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

`void`  
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::XmlWriter
#include <xmlwriter.h>

Inheritance diagram for IO::XmlWriter:

```
IO::XmlWriter
  |
  V
IO::StreamWriter
  |
  V
Core::RefCounted
```
Detailed Description

Write XML-formatted data to a stream.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XmlWriter ()</strong></td>
<td></td>
</tr>
<tr>
<td>Virtual constructor</td>
<td></td>
</tr>
<tr>
<td><strong>~XmlWriter ()</strong></td>
<td></td>
</tr>
<tr>
<td>Virtual destructor</td>
<td></td>
</tr>
<tr>
<td><strong>Open ()</strong></td>
<td>begin writing the stream</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>end writing the stream</td>
</tr>
<tr>
<td>bool <strong>BeginNode (const Util::String &amp;nodeName)</strong></td>
<td>begin a new node under the current node</td>
</tr>
<tr>
<td>void <strong>EndNode ()</strong></td>
<td>end current node, set current node to parent</td>
</tr>
<tr>
<td>void <strong>WriteContent (const Util::String &amp;text)</strong></td>
<td>write content text</td>
</tr>
<tr>
<td>void <strong>WriteComment (const Util::String &amp;comment)</strong></td>
<td>write a comment</td>
</tr>
<tr>
<td>void <strong>SetString (const Util::String &amp;name, const Util::String &amp;value)</strong></td>
<td>set string attribute on current node</td>
</tr>
<tr>
<td>void <strong>.SetBool (const Util::String &amp;name, bool value)</strong></td>
<td>set bool attribute on current node</td>
</tr>
<tr>
<td>void <strong>SetInt (const Util::String &amp;name, int value)</strong></td>
<td>set int attribute on current node</td>
</tr>
<tr>
<td>void <strong>SetFloat (const Util::String &amp;name, float value)</strong></td>
<td>set float attribute on current node</td>
</tr>
<tr>
<td>void <strong>SetFloat2 (const Util::String &amp;name, const Math::float2 &amp;value)</strong></td>
<td>set float2 attribute on current node</td>
</tr>
<tr>
<td>void <strong>SetFloat4 (const Util::String &amp;name, const Math::float4 &amp;value)</strong></td>
<td>set float4 attribute on current node</td>
</tr>
<tr>
<td>void <strong>SetMatrix44 (const Util::String &amp;name, const Math::matrix44 &amp;value)</strong></td>
<td></td>
</tr>
</tbody>
</table>
template<typename T>

```c++
void Set (const Util::String &name, const T &value)
```

generic setter, template specializations implemented in
nebula3/code/addons/nebula2

```c++
void SetStream (const Ptr< Stream > &s)
```

set stream to write to

```c++
const Ptr< Stream > & GetStream () const
```

get currently set stream

```c++
bool HasStream () const
```

return true if a stream is set

```c++
bool IsOpen () const
```

return true if currently open

```c++
int GetRefCount () const
```

get the current refcount

```c++
void AddRef ()
```

increment refcount by one

```c++
void Release ()
```

decrement refcount and destroy object if refcount is zero

```c++
bool IsInstanceOf (const Rtti &rtti) const
```

return true if this object is instance of given class

```c++
bool IsInstanceOf (const Util::String &className) const
```

return true if this object is instance of given class by string

```c++
bool IsInstanceOf (const Util::FourCC &classFourCC) const
```

return true if this object is instance of given class by fourcc

```c++
bool IsA (const Rtti &rtti) const
```

return true if this object is instance of given class, or a derived class

```c++
bool IsA (const Util::String &rttiName) const
```

return true if this object is instance of given class, or a derived class, by string

```c++
bool IsA (const Util::FourCC &rttiFourCC) const
```

return true if this object is instance of given class, or a derived class, by fourcc

```c++
const Util::String & GetClassName () const
```

get the class name
<table>
<thead>
<tr>
<th><code>Util::FourCC</code></th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
</table>

*get the class FourCC code*
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool IO::XmlWriter::Open ( ) [virtual]

begin writing the stream

Open the XML stream for writing. This will create a new TiXmlDocument object which will be written to the stream in Close().

Reimplemented from IO::StreamWriter.

void IO::XmlWriter::Close ( ) [virtual]

end writing the stream

Close the XML stream.

Reimplemented from IO::StreamWriter.

bool IO::XmlWriter::BeginNode ( const Util::String & nodeName )

begin a new node under the current node

Begin a new node. The new node will be set as the current node. Nodes may form a hierarchy. Make sure to finalize a node with a corresponding call to EndNode()!

void IO::XmlWriter::EndNode ( )

end current node, set current node to parent

Finalize current node. This will set the parent of the current node as new current node so that correct hierarchical behaviour is implemented.
void IO::XmlWriter::WriteContent ( const Util::String text & )

write content text
Write inline text at current position.

void IO::XmlWriter::WriteComment ( const Util::String comment & )

write a comment
Write a comment into the XML file.

void IO::XmlWriter::SetString ( const Util::String name, const Util::String value & )

set string attribute on current node
Set the provided attribute to a string value.

void IO::XmlWriter::SetBool ( const Util::String name, bool value )

set bool attribute on current node
Set the provided attribute to a bool value.

void IO::XmlWriter::SetInt ( const Util::String name, int value )
set int attribute on current node

Set the provided attribute to an int value.

```cpp
void IO::XmlWriter::SetFloat ( const Util::String & name, float & value )
```

set float attribute on current node

Set the provided attribute to a float value.

```cpp
void IO::XmlWriter::SetFloat4 ( const Util::String & name, const Math::float4 & value )
```

set float4 attribute on current node

Set the provided attribute to a float4 value.

```cpp
void IO::XmlWriter::SetMatrix44 ( const Util::String & name, const Math::matrix44 & value )
```

set matrix44 attribute on current node

Set the provided attribute to a matrix44 value. The stream must be in Write or ReadWrite mode for this.

```cpp
void IO::StreamWriter::SetStream ( const Ptr<Stream> & s ) [inherited]
```

set stream to write to
Attaches the writer to a stream. This will increment the refcount of the stream.

Reimplemented in **Messaging::MessageWriter**.

```cpp
const Ptr< Stream > & IO::StreamWriter::GetStream() const [inherited]
```

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use **HasStream()** to determine if a stream is attached.

```cpp
bool IO::StreamWriter::HasStream() const [inherited]
```

return true if a stream is set

Returns true if a stream is attached to the writer.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String & Core::RefCounted::GetClassName

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
IO::ZipArchive
#include <ziparchive.h>

Inheritance diagram for IO::ZipArchive:

```
Core::RefCounted

IO::ArchiveBase

IO::ZipArchive

IO::Archive
```
Detailed Description

Private helper class for ZipFileSystem to hold per-Zip-archive data. Uses the zlib and the minizip lib for zip file access.

Multithreading: access to zlib archives needs to be serialized. A ZipArchive objects contains a critical section which it will hand down to ZipFileEntry objects.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ZipArchive ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~ZipArchive ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>bool Setup (const URI &amp;uri)</strong></td>
<td>Setup the archive from an URI</td>
</tr>
<tr>
<td><strong>void Discard ()</strong></td>
<td>Discard the archive</td>
</tr>
<tr>
<td><strong>Util::Array&lt; Util::String &gt; ListFiles (const Util::String &amp;dirPathInArchive, const Util::String &amp;pattern) const</strong></td>
<td>List all files in a directory in the archive</td>
</tr>
<tr>
<td><strong>Util::Array&lt; Util::String &gt; ListDirectories (const Util::String &amp;dirPathInArchive, const Util::String &amp;pattern) const</strong></td>
<td>List all subdirectories in a directory in the archive</td>
</tr>
<tr>
<td><strong>URI ConvertToArchiveURI (const URI &amp;fileURI) const</strong></td>
<td>Convert a &quot;file:&quot; URI into a &quot;zip:&quot; URI pointing into this archive</td>
</tr>
<tr>
<td><strong>Util::String ConvertToPathInArchive (const Util::String &amp;absPath) const</strong></td>
<td>Convert an absolute path to local path inside archive, returns empty string if absPath doesn't point into this archive</td>
</tr>
<tr>
<td><strong>bool IsValid () const</strong></td>
<td>Return true if archive is valid</td>
</tr>
<tr>
<td><strong>const URI &amp; GetURI () const</strong></td>
<td>Get the URI of the archive</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;className) const</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

`bool IO::ZipArchive::Setup(const URI & zipFileURI)`

setup the archive from an URI

This opens the zip archive and reads the table of content as a tree of `ZipDirEntry` and `ZipFileEntry` objects.

Reimplemented from `IO::ArchiveBase`.

`void IO::ZipArchive::Discard()`

discard the archive

This closes the zip archive, releasing the table of contents and closing the zip file.

Reimplemented from `IO::ArchiveBase`.

`URI IO::ZipArchive::ConvertToArchiveURI(const URI & fileURI)`

convert a "file:" URI into a "zip:" URI pointing into this archive

This method takes a normal "file:" scheme URI and convertes it into a "zip:" scheme URI which points to the file in this zip archive. This is used by the IoServer for transparent file access into zip archives.

Reimplemented from `IO::ArchiveBase`.

`String IO::ZipArchive::ConvertToPathInArchive(const Util::String absPath)`

convert an absolute path to local path inside archive, returns empty string if absPath doesn't point into this archive
Test if an absolute path points into the zip archive and return a locale path into the zip archive. This will not test, whether the file or directory inside the zip archive actually exists, only if the path points INTO the zip archive by checking against the location directory of the zip archive.

Reimplemented from `IO::ArchiveBase`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::ZipDirEntry
IO::ZipDirEntry Class Reference

#include <zipdirententry.h>
Detailed Description

A directory entry in a zip archive. The ZipDirEntry class is thread-safe, all public methods can be invoked from on the same object from different threads.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZipDirEntry ()</td>
<td>constructor</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; GetName () const</td>
<td>get the name of the dir entry</td>
</tr>
<tr>
<td>ZipFileEntry * FindFileEntry (const Util::StringAtom &amp;name) const</td>
<td>find a direct child file entry, return 0 if not exists</td>
</tr>
<tr>
<td>ZipDirEntry * FindDirEntry (const Util::StringAtom &amp;name) const</td>
<td>find a direct child directory entry, return 0 if not exists</td>
</tr>
<tr>
<td>const Util::Array&lt; ZipDirEntry &gt; &amp; GetDirEntries () const</td>
<td>get directory entries</td>
</tr>
<tr>
<td>const Util::Array&lt; ZipFileEntry &gt; &amp; GetFileEntries () const</td>
<td>get file entries</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:47 2010
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

IO::ZipFileEntry
IO::ZipFileEntry Class Reference

#include <zipfileentry.h>
Detailed Description

A file entry in a zip archive. The ZipFileEntry class is thread-safe, all public methods can be invoked from on the same object from different threads.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ZipFileEntry ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~ZipFileEntry ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>const Util::StringAtom &amp; GetName ()</strong> const</td>
<td>get name of the file entry</td>
</tr>
<tr>
<td><strong>IO::Stream::Size GetFileSize ()</strong> const</td>
<td>get the uncompressed file size in bytes</td>
</tr>
<tr>
<td><strong>bool Open (const Util::String &amp;password=&quot;&quot;)</strong></td>
<td>open the zip file</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close the zip file</td>
</tr>
<tr>
<td>*<em>bool Read (void <em>buf, IO::Stream::Size bufSize)</em></em> const</td>
<td>read the <em>entire</em> content into the provided memory buffer</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::ZipFileStream
#include <zipfilestream.h>

Inheritance diagram for IO::ZipFileStream:
Detailed Description

Wraps a file in a zip archive into a stream. The file int the zip-archive is not cached. Only forward reading is allowed. Only one file must be opened in that archive at a time.

The IO::Server allows transparent access to data in zip files through normal "file:" URIs by first checking whether the file is part of a mounted zip archive. Only if this is not the case, the file will be opened as normal.

To force reading from a zip archive, use an URI of the following format:

zip://[samba server]/bla/blob/archive.zip?
file=path/in/zipfile&pwd=password

This assumes that the URI scheme "zip" has been associated with the ZipFileStream class using the IO::Server::RegisterUriScheme() method.

The server and local path part of the URI contain the path to the zip archive file. The query part contains the path of the file in the zip archive and an optional password.

(C) 2006 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><strong>AccessMode</strong> access modes</td>
</tr>
<tr>
<td>enum</td>
<td><strong>AccessPattern</strong> access preferred pattern</td>
</tr>
<tr>
<td>enum</td>
<td><strong>SeekOrigin</strong> seek origins</td>
</tr>
<tr>
<td>typedef int</td>
<td><strong>Position</strong> typedefs</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ZipFileStream()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~ZipFileStream()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>virtual bool CanRead() const</code></td>
<td>memory streams support reading</td>
</tr>
<tr>
<td><code>virtual bool CanWrite() const</code></td>
<td>memory streams support writing</td>
</tr>
<tr>
<td><code>virtual bool CanSeek() const</code></td>
<td>memory streams support seeking</td>
</tr>
<tr>
<td><code>virtual bool CanBeMapped() const</code></td>
<td>memory streams are mappable</td>
</tr>
<tr>
<td><code>virtual Size GetSize() const</code></td>
<td>get the size of the stream in bytes</td>
</tr>
<tr>
<td><code>virtual Position GetPosition() const</code></td>
<td>get the current position of the read/write cursor</td>
</tr>
<tr>
<td><code>virtual bool Open()</code></td>
<td>open the stream</td>
</tr>
<tr>
<td><code>virtual void Close()</code></td>
<td>close the stream</td>
</tr>
<tr>
<td><code>virtual Size Read(void *ptr, Size numBytes)</code></td>
<td>directly read from the stream</td>
</tr>
<tr>
<td><code>virtual void Seek(Offset offset, SeekOrigin origin)</code></td>
<td>seek in stream, only forward seeks are allowed</td>
</tr>
<tr>
<td><code>virtual bool Eof() const</code></td>
<td>return true if end-of-stream reached</td>
</tr>
<tr>
<td><code>virtual void * Map()</code></td>
<td>map for direct memory-access</td>
</tr>
<tr>
<td><code>virtual void Unmap()</code></td>
<td>unmap a mapped stream</td>
</tr>
<tr>
<td><code>void SetURI(const URI &amp;u)</code></td>
<td>set stream location as URI</td>
</tr>
<tr>
<td><code>const URI &amp; GetURI()</code></td>
<td>const URI &amp;</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>virtual void SetSize(Size s)</code></td>
<td>set a new size for the stream</td>
</tr>
<tr>
<td><code>void SetAccessMode(AccessMode m)</code></td>
<td>set the access mode of the stream (default is ReadAccess)</td>
</tr>
<tr>
<td><code>AccessMode GetAccessMode()</code></td>
<td>get the access mode of the stream</td>
</tr>
<tr>
<td><code>void SetAccessPattern(AccessPattern p)</code></td>
<td>set the preferred access pattern (default is Sequential)</td>
</tr>
<tr>
<td><code>AccessPattern GetAccessPattern()</code></td>
<td>get the preferred access pattern</td>
</tr>
<tr>
<td><code>void SetMediaType(const MediaType &amp;t)</code></td>
<td>set optional media type of stream content</td>
</tr>
<tr>
<td><code>const MediaType &amp; GetMediaType()</code></td>
<td>get optional media type</td>
</tr>
<tr>
<td><code>bool IsOpen()</code></td>
<td>return true if currently open</td>
</tr>
<tr>
<td><code>virtual void Write(const void *ptr, Size numBytes)</code></td>
<td>directly write to the stream</td>
</tr>
<tr>
<td><code>virtual Size Read(void *ptr, Size numBytes)</code></td>
<td>directly read from the stream</td>
</tr>
<tr>
<td><code>virtual void Seek(Offset offset, SeekOrigin origin)</code></td>
<td>seek in stream</td>
</tr>
<tr>
<td><code>virtual void Flush()</code></td>
<td>flush unsaved data</td>
</tr>
<tr>
<td><code>bool IsMapped()</code></td>
<td>return true if stream is currently mapped to memory</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>Get the class FourCC code</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>static void DumpRefCountingLeaks ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool IO::ZipFileStream::Open( ) [virtual]

open the stream

Open the stream for reading. This will decompress the entire file from the zip archive into memory.

Reimplemented from IO::Stream.

void IO::Stream::SetURI(const URI& u) [inherited]

set stream location as URI

Set the URI of the stream as string. The URI identifies the source resource of the stream.

const URI& IO::Stream::GetURI( ) const [inherited]

get stream URI

Get the URI of the stream as string.

void IO::Stream::SetSize(Size s) [virtual, inherited]

set a new size for the stream

This sets a new size for the stream. Not all streams support this method. If the new size if smaller then the existing size, the contents will be clipped.

void IO::Stream::SetAccessMode(AccessMode m) [inherited]

set the access mode of the stream (default is ReadAccess)
This method sets the intended access mode of the stream. The actual behaviour depends on the implementation of the derived class. The default is ReadWrite.

**Stream::AccessMode**

```cpp
IO::Stream::GetAccessMode ( ) const [inherited]
```

get the access mode of the stream

Get the access mode of the stream.

```cpp
void IO::Stream::SetAccessPattern ( AccessPattern p ) [inherited]
```

set the preferred access pattern (default is Sequential)

Set the preferred access pattern of the stream. This can be Random or Sequential. This is an optional flag to improve performance with some stream implementations. The default is sequential. The pattern cannot be changed while the stream is open.

**Stream::AccessPattern**

```cpp
IO::Stream::GetAccessPattern ( ) const [inherited]
```

get the preferred access pattern

Get the currently set preferred access pattern of the stream.

```cpp
bool IO::Stream::IsOpen ( ) const [inherited]
```

return true if currently open

Return true if the stream is currently open.

```cpp
void IO::Stream::Write ( const void * ptr, Size numBytes ) [virtual, inherited]
```

directly write to the stream
Write raw data to the stream. For more convenient writing, attach the stream to an \texttt{IO::StreamWriter} object. This method is only valid if the stream class returns true in \texttt{CanWrite()}.

\begin{verbatim}
void Stream::Size

IO::Stream::Read ( 
   * ptr,
   Size numBytes
)
\end{verbatim}

directly read from the stream

Read raw data from the stream. For more convenient reading, attach the stream to an \texttt{IO::StreamReader} object. The method returns the number of bytes actually read. This method is only valid if the stream class returns true in \texttt{CanRead()}. Returns the number of bytes actually read from the stream, this may be less than numBytes, or 0 if end-of-stream is reached.

\begin{verbatim}
void IO::Stream::Seek ( 
   Offset offset,
   SeekOrigin origin
)
\end{verbatim}

seek in stream

Move the read/write cursor to a new position, returns the new position in the stream. This method is only supported if the stream class returns true in \texttt{CanSeek()}.

\begin{verbatim}
void IO::Stream::Flush ( )
\end{verbatim}

flush unsaved data

Flush any unsaved data. Note that unsaved data will also be flushed automatically when the stream is closed.

Reimplemented in \texttt{IO::FileStream}.

\begin{verbatim}
bool IO::Stream::IsMapped ( ) const
\end{verbatim}
return true if stream is currently mapped to memory

Returns true if the stream is currently mapped.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrease refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::ZipFileSystem
#include <zipfilesystem.h>

Inheritance diagram for IO::ZipFileSystem:

```
IO::ZipFileSystem

IO::ArchiveFileSystem
```

IO::ZipFileSystem Class Reference
Detailed Description

An archive filesystem wrapper for ZIP files.

Uses the zlib and the minizip package under the hood.

Limitations: No write access. No seek on compressed data, the ZipFileSystem will generally decompress an entire file into memory at once, so that the ZipStreamClass can provide random access on the decompressed data. Thus the typical "audio streaming scenario" is not possible from zip files (that's what XACT's sound banks is there for anyway :)

How to fix the no-seek problem: zlib processes datas in chunks, and cannot seek randomly within a chunk, and the chunk size is dependent on the compress application being used to create the zip file(?), if those internals are known, it would be possible to write a chunked filesystem which keeps buffered chunks around for each client, probably not worth the effort. Another appoach would be to split stream-files into "chunk-files" before compressing, and to read the next complete chunk files when new data is needed. This approach doesn't require changes to the strip filesystem.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ZipFileSystem ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~ZipFileSystem ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>Setup ()</strong></td>
<td>setup the archive file system</td>
</tr>
<tr>
<td>void <strong>Discard ()</strong></td>
<td>discard the archive file system</td>
</tr>
<tr>
<td>**Ptr&lt; Archive &gt; <strong>FindArchiveWithFile (const URI &amp;fileUri) const</strong></td>
<td>find first archive which contains the file path</td>
</tr>
<tr>
<td>**Ptr&lt; Archive &gt; <strong>FindArchiveWithDir (const URI &amp;dirUri) const</strong></td>
<td>find first archive which contains the directory path</td>
</tr>
</tbody>
</table>
void
IO::ZipFileSystem::Setup()

setup the archive file system

Setup the **ZipFileSystem**. Registers the **ZipFileStream** class.

```cpp
Ptr< Archive >
IO::ZipFileSystem::FindArchiveWithFile( const URI & uri ) const
```

find first archive which contains the file path

This method takes a normal file **URI** and checks if the local path of the **URI** is contained as file entry in any mounted zip archive. If yes ptr to the zip archive is returned, otherwise a 0 pointer. NOTE: if the same path resides in several zip archives, it is currently not defined which one will be returned (the current implementation returns the first zip archive in alphabetical order which contains the file).

```cpp
Ptr< Archive >
IO::ZipFileSystem::FindArchiveWithDir( const URI & uri ) const
```

find first archive which contains the directory path

Same as **FindArchiveWithFile()**, but checks for a directory entry in a zip file.
#include <job.h>

Inheritance diagram for Jobs::Job:
Detailed Description

A Job in the Nebula3 job system.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void Setup</code></td>
<td>(const <code>Jobs::JobUniformDesc</code> &amp;uniformDesc, const <code>Jobs::JobDataDesc</code> &amp;inputDesc, const <code>Jobs::JobDataDesc</code> &amp;outputDesc, const <code>Jobs::JobFuncDesc</code> &amp;funcDesc) setup the job</td>
</tr>
<tr>
<td><code>void Discard</code></td>
<td>() discard the job</td>
</tr>
<tr>
<td><code>void * AllocPrivateBuffer</code></td>
<td>(Memory::HeapType heapType, SizeT size) allocate a single memory buffer associated with the job, may only be called once, and before <code>Setup()</code>!</td>
</tr>
<tr>
<td><code>void * GetPrivateBuffer</code></td>
<td>() const get pointer to job's optional private buffer</td>
</tr>
<tr>
<td><code>SizeT GetPrivateBufferSize</code></td>
<td>() const get size of job's optional private buffer</td>
</tr>
<tr>
<td><code>bool IsValid</code></td>
<td>() const return true if job has been setup</td>
</tr>
<tr>
<td><code>void PatchInputDesc</code></td>
<td>(const <code>Jobs::JobDataDesc</code> &amp;inputDesc) patch input pointers after job has been setup</td>
</tr>
<tr>
<td><code>void PatchOutputDesc</code></td>
<td>(const <code>Jobs::JobDataDesc</code> &amp;outputDesc) patch output pointers after job has been setup</td>
</tr>
<tr>
<td><code>void PatchUniformDesc</code></td>
<td>(const <code>Jobs::JobUniformDesc</code> &amp;uniformDesc) patch uniform pointer after job has been setup</td>
</tr>
<tr>
<td><code>const Jobs::JobUniformDesc &amp; GetUniformDesc</code></td>
<td>() const get uniform data descriptor</td>
</tr>
<tr>
<td><code>const Jobs::JobDataDesc &amp; GetInputDesc</code></td>
<td>() const get input data descriptor</td>
</tr>
<tr>
<td><code>const Jobs::JobDataDesc &amp; GetOutputDesc</code></td>
<td>() const get output data descriptor</td>
</tr>
<tr>
<td><code>const Jobs::JobFuncDesc &amp; GetFuncDesc</code></td>
<td>() const</td>
</tr>
<tr>
<td>get function descriptor</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td></td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td></td>
</tr>
<tr>
<td>get the current refcount</td>
<td></td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td></td>
</tr>
<tr>
<td>increment refcount by one</td>
<td></td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td></td>
</tr>
<tr>
<td>decrement refcount and destroy object if refcount is zero</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td></td>
</tr>
<tr>
<td>return true if this object is instance of given class</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td></td>
</tr>
<tr>
<td>return true if this object is instance of given class by string</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
<td></td>
</tr>
<tr>
<td>return true if this object is instance of given class by fourcc</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td></td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
<td></td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
<td></td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetClassName</strong> () const</td>
<td></td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td><strong>Util::FourCC</strong> <strong>GetClassFourCC</strong> () const</td>
<td></td>
</tr>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Static Public Attributes

\[
\text{static const SizeT } \textbf{MaxSliceSize} = \textbf{JobMaxSliceSize}
\]

*max size of a data slice is 16 kByte - 1 byte*
Allocate a single memory buffer associated with the job. The method must be called before Setup(), and will remain valid until the destructor of the job object is called (so it will survive a Discard()).

```c++
void * Base::JobBase::AllocPrivateBuffer ( Memory::HeapType heapType,
                                           SizeT size ) [inherited]
```

Allocate a single memory buffer associated with the job, may only be called once, and before Setup()!

```c++
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

Get the current refcount

Return the current refcount of the object.

```c++
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

Increment refcount by one

Increment the refcount of the object.

```c++
void Core::RefCounted::Release ( ) [inline, inherited]
```

Decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```c++
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

Get the class name
Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

---

The Nebula Device 3 documentation generated by `doxygen` at Fri Mar 26 15:21:47 2010
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Jobs:: \texttt{JobContext}
Jobs::JobContext Class Reference

#include <jobfunccontext.h>
Detailed Description

**JobContext** structure handed to job processing functions. Note that this header must be self-contained, since it will be included on the SPU side on PS3.

(C) 2009 Radon Labs GmbH
| Home | Namespace List |

- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**Jobs**: JobDataDesc
Jobs::JobDataDesc Class Reference

#include <jobdatadesc.h>
Detailed Description

Descriptor for input/output data of a job. Input/output data is split into elements and slices. A job function may be called with any number of elements, up to the MaxElementsPerSlice number. Within a current slice, the job may perform random access on elements. Slices may not depend on each other (the job system may split a job into slices which are processed in parallel).

(C) 2009 Radon Labs GmbH
### Public Member Functions

**JobDataDesc ()**
*default constructor*

**JobDataDesc (void *ptr, SizeT bufSize, SizeT sliceSize)**
*constructor with 1 data buffer*

**JobDataDesc (void *ptr0, SizeT bufSize0, SizeT sliceSize0, void *ptr1, SizeT bufSize1, SizeT sliceSize1)**
*constructor with 2 data buffers*

**JobDataDesc (void *ptr0, SizeT bufSize0, SizeT sliceSize0, void *ptr1, SizeT bufSize1, SizeT sliceSize1, void *ptr2, SizeT bufSize2, SizeT sliceSize2)**
*constructor with 3 data buffers*

**JobDataDesc (void *ptr0, SizeT bufSize0, SizeT sliceSize0, void *ptr1, SizeT bufSize1, SizeT sliceSize1, void *ptr2, SizeT bufSize2, SizeT sliceSize2, void *ptr3, SizeT bufSize3, SizeT sliceSize3)**
*constructor with 4 data buffers*

**void Update (IndexT index, void *ptr, SizeT bufSize, SizeT sliceSize)**
*update a parameter set*

**SizeT GetNumBuffers () const**
*get number of buffers*

**void * GetPointer (IndexT i) const**
*get buffer pointer*

**SizeT GetBufferSize (IndexT i) const**
*get buffer size*

**SizeT GetSliceSize (IndexT i) const**
*get slice size*
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Jobs:: JobFuncDesc
Jobs::JobFuncDesc Class Reference

#include <jobfuncdesc.h>
Detailed Description

Platform-wrapper for a Job function descriptor.

(C) 2009 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Jobs:: JobPort
# Jobs::JobPort Class Reference

```c
#include <jobport.h>
```

Inheritance diagram for Jobs::JobPort:

```
Core::RefCounted

Base::JobPortBase

Jobs::TIPJobPort

Jobs::JobPort
```
Detailed Description

A job port of the N3 job system (see JobPortBase for details).

(C) 2009 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>void</th>
<th><strong>Discard</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>discard the job port</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>PushJob</strong> (const <strong>Ptr&lt; Job &gt;</strong> &amp;job)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>push a job for execution</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>PushJobChain</strong> (const <strong>Util::Array&lt; Ptr&lt; Jobs::Job &gt;</strong> &gt; &amp;jobs)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>push a job chain, each job in the chain depends on previous job</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>PushFlush</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>push a flush command (no effect in thread-pool job system)</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>PushSync</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>push a sync command</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>WaitDone</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>wait for completion</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>CheckDone</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>check for completion, return immediately</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>Setup</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>setup the job port</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsValid</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>return true if the job object is valid</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><strong>GetRefCount</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>get the current refcount</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>AddRef</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>increment refcount by one</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>Release</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>decrement refcount and destroy object if refcount is zero</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>return true if this object is instance of given class</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>return true if this object is instance of given class by string</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>return true if this object is instance of given class by fourcc</em></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by string</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by fourcc</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>Util::String</strong> &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
</table>

*dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)*
**Member Function Documentation**

```cpp
void Jobs::TPJobPort::PushJobChain(const Util::Array<Ptr< Jobs::Job >> & jobs ) [inherited]
```

Push a job chain, where each job in the chain depends on the previous job. This will also guarantee, that the first job slice of each job will run on the same worker thread. In case of simple jobs (with only one slice) this improves load.

Reimplemented from `Base::JobPortBase`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

Get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

Increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

```
Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```
void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Jobs:: **JobServerBase**
Jobs::JobServerBase Class Reference

#include <jobsystembase.h>
Detailed Description

The **JobSystem** singleton is used to setup and shutdown the Jobs subsystem.

(C) 2009 Radon Labs GmbH
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Jobs:: JobSystem
#include <jobsystem.h>

Inheritance diagram for Jobs::JobSystem:
Detailed Description

Initializes the N3 job system.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobSystem ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~JobSystem ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>void Setup ()</td>
<td>Setup the job system</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>Shutdown the job system</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:47 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Jobs:: JobUniformDesc
Jobs::JobUniformDesc Class Reference

#include <jobuniformdesc.h>
Detailed Description

Descriptor for uniform data of a Job.

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobUniformDesc ()</code></td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><code>JobUniformDesc (void *ptr, SizeT bufSize, SizeT scratchSize)</code></td>
<td><em>constructor with 1 uniform buffer</em></td>
</tr>
<tr>
<td><code>JobUniformDesc (void *ptr0, SizeT bufSize0, void *ptr1, SizeT bufSize1, SizeT scratchSize)</code></td>
<td><em>constructor with 2 uniform buffers</em></td>
</tr>
<tr>
<td>void <strong>Update</strong> *(IndexT index, void <em>ptr, SizeT bufSize, SizeT scratchSize)</em></td>
<td><em>update the uniform desc, all sizes in bytes</em></td>
</tr>
<tr>
<td>SizeT <strong>GetNumBuffers () const</strong></td>
<td><em>get number of buffers</em></td>
</tr>
<tr>
<td>void * <strong>GetPointer</strong> <em>(IndexT i) const</em></td>
<td><em>get buffer pointer</em></td>
</tr>
<tr>
<td>SizeT <strong>GetBufferSize</strong> <em>(IndexT i) const</em></td>
<td><em>get buffer size</em></td>
</tr>
<tr>
<td>SizeT <strong>Get ScratchSize () const</strong></td>
<td><em>get scratch size</em></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:47 2010
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Jobs:: SerialJob
#include <serialjob.h>

Inheritance diagram for Jobs::SerialJob:

```
Core::RefCounted

Base::JobBase

Jobs::SerialJob
```
Detailed Description

A job in the serial job system (dummy job system which doesn't parallelize jobs).

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SerialJob ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~SerialJob ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>AllocPrivateBuffer (Memory::HeapType heapType, SizeT size)</strong></td>
<td>allocate a single memory buffer associated with the job, may only be called once, and before <code>Setup()</code>!</td>
</tr>
<tr>
<td><strong>GetPrivateBuffer () const</strong></td>
<td>get pointer to job's optional private buffer</td>
</tr>
<tr>
<td><strong>GetPrivateBufferSize () const</strong></td>
<td>get size of job's optional private buffer</td>
</tr>
<tr>
<td><strong>Setup (const Jobs::JobUniformDesc &amp;uniformDesc, const Jobs::JobDataDesc &amp;inputDesc, const Jobs::JobDataDesc &amp;outputDesc, const Jobs::JobFuncDesc &amp;funcDesc)</strong></td>
<td>setup the job</td>
</tr>
<tr>
<td><strong>Discard ()</strong></td>
<td>discard the job</td>
</tr>
<tr>
<td><strong>IsValid () const</strong></td>
<td>return true if job has been setup</td>
</tr>
<tr>
<td><strong>PatchInputDesc (const Jobs::JobDataDesc &amp;inputDesc)</strong></td>
<td>patch input pointers after job has been setup</td>
</tr>
<tr>
<td><strong>PatchOutputDesc (const Jobs::JobDataDesc &amp;outputDesc)</strong></td>
<td>patch output pointers after job has been setup</td>
</tr>
<tr>
<td><strong>PatchUniformDesc (const Jobs::JobUniformDesc &amp;uniformDesc)</strong></td>
<td>patch uniform pointer after job has been setup</td>
</tr>
<tr>
<td><strong>GetUniformDesc () const</strong></td>
<td>get uniform data descriptor</td>
</tr>
<tr>
<td><strong>GetInputDesc () const</strong></td>
<td></td>
</tr>
</tbody>
</table>
get input data descriptor

const Jobs::JobDataDesc & GetOutputDesc () const
get output data descriptor

const Jobs::JobFuncDesc & GetFuncDesc () const
get function descriptor

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrease refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Static Public Attributes

```c
static const SizeT MaxSliceSize = JobMaxSliceSize
```

*max size of a data slice is 16 kByte - 1 byte*
allocate a single memory buffer associated with the job, may only be called once, and before Setup()!

This method can be used to allocate a single memory buffer, owned by the job object. The method must be called before Setup(), and will remain valid until the destructor of the job object is called (so it will survive a Discard())!

get the current refcount

Return the current refcount of the object.

increment refcount by one

Increment the refcount of the object.

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

get the class name
Get the class name of the object.

`Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

`void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Jobs:: SerialJobFuncDesc
Jobs::SerialJobFuncDesc Class Reference

#include <serialjobfuncdesc.h>

Inheritance diagram for Jobs::SerialJobFuncDesc:
Detailed Description

A function pointer descriptor for the dummy job system. This is compatible with the JobFuncDesc from the TP (thread-pool) job system.

(C) 2009 Radon Labs GmbH
Public Types

typedef void(*) `FuncPtr` (const JobFuncContext &ctx)

callback function typedef
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerialJobFuncDesc ()</td>
<td>default constructor</td>
</tr>
<tr>
<td>SerialJobFuncDesc (FuncPtr funcPtr)</td>
<td>constructor</td>
</tr>
<tr>
<td>FuncPtr GetFunctionPointer () const</td>
<td>get function pointer</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:47 2010
Jobs::**SerialJobPort**
#include <serialjobport.h>

Inheritance diagram for Jobs::SerialJobPort:

```
Core::RefCounted

Base::JobPortBase

Jobs::SerialJobPort
```
Detailed Description

JobPort implementation in the serial job system. In the serial job system, PushJob() will immediately execute the job and return after the job has executed. All other methods are empty and return immediately.

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SerialJobPort()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~SerialJobPort()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>void PushJob(const Ptr&lt; Jobs::Job &gt; &amp;job)</code></td>
<td>push a job for execution</td>
</tr>
<tr>
<td><code>void PushJobChain(const Util::Array&lt; Ptr&lt; Jobs::Job &gt; &gt; &amp;jobs)</code></td>
<td>push a job chain, each job in the chain depends on previous job</td>
</tr>
<tr>
<td><code>void PushFlush()</code></td>
<td>push a flush command (makes sure that jobs don't re-use uniform data from previous jobs)</td>
</tr>
<tr>
<td><code>void PushSync()</code></td>
<td>push a sync command (waits for completion of all previous jobs on this port)</td>
</tr>
<tr>
<td><code>void WaitDone()</code></td>
<td>wait for completion</td>
</tr>
<tr>
<td><code>bool CheckDone()</code></td>
<td>check for completion, return immediately</td>
</tr>
<tr>
<td><code>void Setup()</code></td>
<td>setup the job port</td>
</tr>
<tr>
<td><code>void Discard()</code></td>
<td>discard the job port</td>
</tr>
<tr>
<td><code>bool IsValid() const</code></td>
<td>return true if the job object is valid</td>
</tr>
<tr>
<td><code>int GetRefCount() const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>const</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>return true if this object is instance of given class by string</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class by fourcc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>Isa (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class, or a derived class</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>Isa (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class, or a derived class, by string</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>Isa (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class name</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Jobs::SerialJobPort::PushJob(
    const Ptr<Jobs::Job> & job)
```

push a job for execution

This executes the job immediately in the current thread and returns afterwards.

Reimplemented from `Base::JobPortBase`.

```cpp
void Jobs::SerialJobPort::PushJobChain(
    const Util::Array<Ptr<Jobs::Job>> & jobs)
```

push a job chain, each job in the chain depends on previous job

Push a job chain, where each job in the chain depends on the previous job. This will also guarantee, that the first job slice of each job will run on the same worker thread. In case of simple jobs (with only one slice) this improves load.

Reimplemented from `Base::JobPortBase`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Jobs::SerialJobSystem
Jobs::SerialJobSystem Class Reference

#include <serialjobsystem.h>
Detailed Description

A dummy job system which processes jobs in the caller thread. This is useful for single-core systems like the Wii.

(C) 2009 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerialJobSystem ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~SerialJobSystem ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void Setup ()</td>
<td>setup the job system</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>shutdown the job system</td>
</tr>
</tbody>
</table>
Jobs: TPJob
#include <tpjob.h>

Inheritance diagram for Jobs::TPJob:
Detailed Description

Job implementation for the thread-pool job system.

(C) 2009 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TPJob ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~TPJob ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Setup</strong> (const Jobs::JobUniformDesc &amp;uniformDesc, const Jobs::JobDataDesc &amp;inputDesc, const Jobs::JobDataDesc &amp;outputDesc, const Jobs::JobFuncDesc &amp;funcDesc)</td>
<td>setup the job</td>
</tr>
<tr>
<td><strong>Discard ()</strong></td>
<td>discard the job</td>
</tr>
<tr>
<td><strong>AllocPrivateBuffer</strong> (Memory::HeapType heapType, SizeT size)</td>
<td>allocate a single memory buffer associated with the job, may only be called once, and before Setup();</td>
</tr>
<tr>
<td><strong>GetPrivateBuffer</strong> () const</td>
<td>get pointer to job's optional private buffer</td>
</tr>
<tr>
<td><strong>GetPrivateBufferSize</strong> () const</td>
<td>get size of job's optional private buffer</td>
</tr>
<tr>
<td><strong>IsValid</strong> () const</td>
<td>return true if job has been setup</td>
</tr>
<tr>
<td><strong>PatchInputDesc</strong> (const Jobs::JobDataDesc &amp;inputDesc)</td>
<td>patch input pointers after job has been setup</td>
</tr>
<tr>
<td><strong>PatchOutputDesc</strong> (const Jobs::JobDataDesc &amp;outputDesc)</td>
<td>patch output pointers after job has been setup</td>
</tr>
<tr>
<td><strong>PatchUniformDesc</strong> (const Jobs::JobUniformDesc &amp;uniformDesc)</td>
<td>patch uniform pointer after job has been setup</td>
</tr>
<tr>
<td><strong>GetUniformDesc</strong> () const</td>
<td>get uniform data descriptor</td>
</tr>
<tr>
<td><strong>GetInputDesc</strong> () const</td>
<td></td>
</tr>
</tbody>
</table>
get input data descriptor

const Jobs::JobDataDesc & GetOutputDesc () const
get output data descriptor

const Jobs::JobFuncDesc & GetFuncDesc () const
get function descriptor

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
derecrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Static Public Attributes

\[
\text{static const SizeT } \textbf{MaxSliceSize} = \textbf{JobMaxSliceSize}
\]

max size of a data slice is 16 kByte - 1 byte
**Member Function Documentation**

```cpp
void * Base::JobBase::AllocPrivateBuffer ( Memory::HeapType heapType,
                                   SizeT size
                                   ) [inherited]
```

allocate a single memory buffer associated with the job, may only be called once, and before **Setup()**!

This method can be used to allocate a single memory buffer, owned by the job object. The method must be called before **Setup()**, and will remain valid until the destructor of the job object is call (so it will survive a **Discard()**).

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

get the class name
Get the class name of the object.

*Util::FourCC*

Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void

Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Jobs:: TPJobCommand
Jobs::TPJobCommand Class Reference

#include <tpjobcommand.h>
Detailed Description

A command queue entry for the worker threads.

(C) 2009 Radon Labs GmbH
Public Types

enum **Code**

*commands*
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TPJobCommand</strong></td>
<td>()\nconstructor</td>
</tr>
<tr>
<td>void <strong>SetupSync</strong></td>
<td>(const Threading::Event *syncEvent)\nsetup for sync command</td>
</tr>
<tr>
<td>void <strong>SetupRun</strong></td>
<td>(TPJobSlice *firstSlice, ushort numSlices, ushort stride)\nsetup for run job slices command</td>
</tr>
<tr>
<td><strong>GetCode</strong></td>
<td>() const\nget the command code</td>
</tr>
<tr>
<td>const Threading::Event * <strong>GetSyncEvent</strong></td>
<td>() const\nget pointer to sync event object</td>
</tr>
<tr>
<td>TPJobSlice * <strong>GetFirstSlice</strong></td>
<td>() const\nget pointer to first job slice</td>
</tr>
<tr>
<td>ushort <strong>GetNumSlices</strong></td>
<td>() const\nget number of slices</td>
</tr>
<tr>
<td>ushort <strong>GetStride</strong></td>
<td>() const\nget slice array stride</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:47 2010
Jobs: `TPJobFuncDesc`
# Jobs::TPJobFuncDesc Class Reference

```c
#include <tpjobfuncdesc.h>
```

Inheritance diagram for Jobs::TPJobFuncDesc:

```
  Base::JobFuncDescBase
   
  Jobs::TPJobFuncDesc
```

Detailed Description

A function descriptor for the thread-pool job system.

(C) 2009 Radon Labs GmbH
Public Types

typedef void(*) FuncPtr (const JobFuncContext &ctx)

callback function typedef
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TPJobFuncDesc()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>TPJobFuncDesc(FuncPtr funcPtr)</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>FuncPtr GetFunctionPointer()</code> const</td>
<td>get function pointer</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:47 2010
Jobs:: TPJobPort
Jobs::TPJobPort Class Reference

#include <tpjobport.h>

Inheritance diagram for Jobs::TPJobPort:
Detailed Description

Thread-pool implementation of JobPort.

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TPJobPort ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~TPJobPort ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>Discard the job port</td>
</tr>
<tr>
<td>void PushJob (const Ptr&lt; Job &gt; &amp;job)</td>
<td>Push a job for execution</td>
</tr>
<tr>
<td>void PushJobChain (const Util::Array&lt; Ptr&lt; Jobs::Job &gt; &gt; &amp;jobs)</td>
<td>Push a job chain, each job in the chain depends on previous job</td>
</tr>
<tr>
<td>void PushFlush ()</td>
<td>Push a flush command (no effect in thread-pool job system)</td>
</tr>
<tr>
<td>void PushSync ()</td>
<td>Push a sync command</td>
</tr>
<tr>
<td>void WaitDone ()</td>
<td>Wait for completion</td>
</tr>
<tr>
<td>bool CheckDone ()</td>
<td>Check for completion, return immediately</td>
</tr>
<tr>
<td>void Setup ()</td>
<td>Setup the job port</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>Return true if the job object is valid</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class,</td>
</tr>
<tr>
<td></td>
<td>by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class,</td>
</tr>
<tr>
<td></td>
<td>by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
push a job chain, each job in the chain depends on previous job

Push a job chain, where each job in the chain depends on the previous job. This will also guarantee, that the first job slice of each job will run on the same worker thread. In case of simple jobs (with only one slice) this improves load.

Reimplemented from Base::JobPortBase.

get the current refcount

Return the current refcount of the object.

increment refcount by one

Increment the refcount of the object.

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
get the class name

Get the class name of the object.

`Util::FourCC`  
`Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]`

get the class FourCC code

Get the class FourCC of the object.

`void`  
`Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]`

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Jobs:: TPJobSlice
Jobs::TPJobSlice Class Reference

#include <tpjobslice.h>
Detailed Description

A "mini job" which works on a single job slice.

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPJobSlice ()</td>
<td>constructor</td>
</tr>
<tr>
<td>~TPJobSlice ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void Setup (TPJob *job, IndexT sliceIndex)</td>
<td>setup the job slice</td>
</tr>
<tr>
<td>TPJob * GetJob () const</td>
<td>get pointer to job</td>
</tr>
<tr>
<td>IndexT GetSliceIndex () const</td>
<td>get slice index</td>
</tr>
</tbody>
</table>
• Main Page
• Namespaces
• Data Structures
• Files
• Related Pages

• Alphabetical List
• Data Structures
• Class Hierarchy
• Data Fields

Jobs: :TPJobSystem
# Jobs::TPJobSystem Class Reference

```cpp
#include <tpjobsystem.h>
```

Inheritance diagram for Jobs::TPJobSystem:

```
Jobs::TPJobSystem

Jobs::JobSystem
```

---
Detailed Description

Implementation of JobSystem for jobs running in a CPU thread pool.

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPJobSystem ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~TPJobSystem ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void Setup ()</td>
<td>setup the job system</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>shutdown the job system</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Jobs: TPJobThreadPool
Jobs::TPJobThreadPool Class Reference

#include <tpjobthreadpool.h>
Detailed Description

Manages the thread-pool, distributes TPJobSlice objects to the worker threads.

FIXME: class is currently not thread-safe!

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TPJobThreadPool</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~TPJobThreadPool</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Setup ()</strong></td>
<td>setup the thread pool</td>
</tr>
<tr>
<td><strong>void Discard ()</strong></td>
<td>discard the thread pool</td>
</tr>
<tr>
<td><strong>bool IsValid () const</strong></td>
<td>return true if object is setup</td>
</tr>
<tr>
<td>*<em>void PushSync (const Threading::Event <em>syncEvent)</em></em></td>
<td>push a sync command into the thread pool</td>
</tr>
<tr>
<td>*<em>void PushJobSlices (TPJobSlice <em>firstSlice, SizeT numSlices, IndexT threadIndex=InvalidIndex)</em></em></td>
<td>push job slices into the the thread pool</td>
</tr>
<tr>
<td><strong>IndexT GetNextThreadId () const</strong></td>
<td>get a suitable worker thread index (optimally a thread with currently no load)</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Jobs: *TPWorkerThread*
Jobs::TPWorkerThread Class Reference

#include <tpworkerthread.h>

Inheritance diagram for Jobs::TPWorkerThread:
Detailed Description

The worker thread class of the thread-pool job system.

(C) 2009 Radon Labs GmbH
<table>
<thead>
<tr>
<th>enum</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>thread priorities</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TPWorkerThread ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~TPWorkerThread ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>EmitWakeupSignal ()</strong></td>
<td>Called if thread needs a wakeup call before stopping</td>
</tr>
<tr>
<td>virtual void <strong>DoWork ()</strong></td>
<td>This method runs in the thread context</td>
</tr>
<tr>
<td>void <strong>Stop ()</strong></td>
<td>Request threading code to stop, returns when thread has actually finished</td>
</tr>
<tr>
<td>void <strong>PushJobCommand (const TPJobCommand &amp;cmd)</strong></td>
<td>Push a job command onto the job queue</td>
</tr>
<tr>
<td>void <strong>SetPriority (Priority p)</strong></td>
<td>Set the thread priority</td>
</tr>
<tr>
<td><strong>Priority</strong> <strong>GetPriority ()</strong> const</td>
<td>Get the thread priority</td>
</tr>
<tr>
<td>void <strong>SetCoreId (System::Cpu::CoreId coreId)</strong></td>
<td>Set cpu core on which the thread should be running</td>
</tr>
<tr>
<td>System::Cpu::CoreId <strong>GetCoreId ()</strong> const</td>
<td>Get the cpu core on which the thread should be running</td>
</tr>
<tr>
<td>void <strong>SetStackSize (SizeT s)</strong></td>
<td>Set stack size in bytes (default is 4 KByte)</td>
</tr>
<tr>
<td><strong>SizeT</strong> <strong>GetStackSize ()</strong> const</td>
<td>Get stack size</td>
</tr>
<tr>
<td>void <strong>SetName (const Util::String &amp;n)</strong></td>
<td>Set thread name</td>
</tr>
<tr>
<td><strong>const Util::String &amp;</strong> <strong>GetName ()</strong> const</td>
<td>Get thread name</td>
</tr>
<tr>
<td>void <strong>Start ()</strong></td>
<td>Start executing the thread code, returns when thread has actually started</td>
</tr>
<tr>
<td><strong>bool</strong> <strong>IsRunning ()</strong> const</td>
<td></td>
</tr>
<tr>
<td>Method/Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>return true if thread has been started</td>
<td></td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> ( ) const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ( )</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong> ( )</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetClassName</strong> ( ) const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC <strong>GetClassFourCC</strong> ( ) const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void YieldThread()</code></td>
<td>yield the thread (gives up current time slice)</td>
</tr>
<tr>
<td><code>static void SetMyThreadName(const char *n)</code></td>
<td>set thread name from within thread context</td>
</tr>
<tr>
<td><code>static const char * GetMyThreadName()</code></td>
<td>obtain name of thread from within thread context</td>
</tr>
<tr>
<td><code>static Threading::ThreadId GetMyThreadId()</code></td>
<td>get the thread ID of this thread</td>
</tr>
<tr>
<td><code>static void DumpRefCountingLeaks()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

bool ThreadStopRequested () const

check if stop is requested, call from DoWork() to see if the thread proc should quit
set thread name

Set the thread's name. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

get thread name

Get the thread's name. This is the vanilla method which returns the name member. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

start executing the thread code, returns when thread has actually started

Start the thread, this creates a `Win32` thread and calls the static ThreadProc, which in turn calls the virtual `DoWork()` class of this object. The method waits for the thread to start and then returns.

return true if thread has been started

Returns true if the thread is currently running.
win360::win360thread::yieldthread

yield the thread (gives up current time slice)

the yield function is empty on win32 and xbox360.

void win360::win360thread::setmythreadname(const char * n ) [static, inherited]

set thread name from within thread context

static method which sets the name of this thread. this is called from within threadproc. the string pointed to must remain valid until the thread is terminated!

const char * win360::win360thread::getmythreadname( ) [static, inherited]

obtain name of thread from within thread context

static method to obtain the current thread name from anywhere in the thread's code.

threading::threadid win360::win360thread::getmythreadid( ) [static, inherited]

get the thread id of this thread

static method which returns the threadid of this thread.

bool win360::win360thread::threadstoprequested( ) const [inline, protected, inherited]

check if stop is requested, call from dowork() to see if the thread proc should quit

if the derived dowork() method is running in a loop it must regularly check if the process wants the thread to terminate by calling threadstoprequested() and simply return if the result is true. this will cause the thread to shut down.
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application
exits.
Legacy::Nvx2StreamReader
Legacy::Nvx2StreamReader Class Reference

#include <nvx2streamreader.h>
Detailed Description

A stream reader which reads legacy nvx2 binary mesh files.

NOTE: this class exists purely for debugging and shouldn't be used in production code!

(C) 2007 Radon Labs GmbH
Lighting::InternalAbstractLightEntity
Lighting::InternalAbstractLightEntity
Class Reference

#include <internalabstractlightentity.h>

Inheritance diagram for Lighting::InternalAbstractLightEntity:
Detailed Description

**Base** class for light sources. Light sources do not directly influence the render pipeline (like manipulating shader variables, etc...). This will be handled by the **LightServer** and **ShadowServer** singletons which may implement platform specific lighting models.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><strong>LinkType</strong> visibility link types</td>
</tr>
<tr>
<td>typedef</td>
<td><strong>Id</strong> a unique id for graphics entities</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InternalAbstractLightEntity()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~InternalAbstractLightEntity()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>LightType::Code GetLightType()</code></td>
<td>get the light type</td>
</tr>
<tr>
<td><code>void SetColor (const Math::float4&amp;)</code></td>
<td>set primary light color</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetColor () const</code></td>
<td>get primary light color</td>
</tr>
<tr>
<td><code>void SetCastShadows (bool b)</code></td>
<td>enable/disable shadow casting</td>
</tr>
<tr>
<td><code>bool GetCastShadows () const</code></td>
<td>get shadow casting flag</td>
</tr>
<tr>
<td><code>void SetProjMapUvOffsetAndScale (const Math::float4 &amp;v)</code></td>
<td>set projection map UV offset and scale ((x)w-&gt;offset, (z)w-&gt;scale)</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetProjMapUvOffsetAndScale</code></td>
<td>get projection map UV offset and scale</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetInvTransform () const</code></td>
<td>get inverse transform (transforms from world to light space)</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetProjTransform () const</code></td>
<td>get light-projection matrix (transforms from light space to light projection space)</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetInvLightProjTransform ()</code></td>
<td>get world-to-light-projection transform (transforms from world to light projection space)</td>
</tr>
<tr>
<td><code>void SetShadowBufferUvOffsetAndScale (const Math::float4 &amp;uvOffset)</code></td>
<td>set shadow buffer uv rectangle (optionally set by light/shadow servers)</td>
</tr>
<tr>
<td><code>GetShadowBufferUvOffsetAndScale</code></td>
<td></td>
</tr>
</tbody>
</table>
const Math::float4 & () const

get shadow buffer uv rectangle

bool GetCastShadowsThisFrame

get cast shadows this frame

void SetCastShadowsThisFrame (val)

set CastShadowsThisFrame

void SetShadowTransform (const Math::matrix44 &val)

set shadow transform

const Math::matrix44 & GetShadowInvTransform ()

get shadow transform

const Math::matrix44 & GetShadowInvLightProjTransform ()

get shadow projection transform

const Math::matrix44 & GetShadowProjTransform ()

get shadow projection transform

float GetShadowIntensity () const

get ShadowIntensity

void SetShadowIntensity (float val)

set ShadowIntensity

virtual void HandleMessage (const Ptr<Messaging::Message> &msg)

handle a message

bool IsActive () const

return true if entity is currently active (is between OnActivate() / OnDeactivate())

bool IsValid () const

return true if entity is current valid (ready rendering)

Id GetId () const

get the graphics entity's unique id

InternalGraphicsEntityType::Code GetType () const

get the entity type

void SetTransform (const Math::matrix44 &m)

set the entity's world space transform

const Math::matrix44 & GetTransform () const
get the entity's world space transform

```cpp
void SetVisible (bool b)
set the entity's visibility

bool IsVisible () const
return true if entity is set to visible

const Ptr< InternalStage > & GetStage () const
get the stage this entity is attached to

bool IsAttachedToStage () const
return true if entity is attached to stage
```

```cpp
Timing::Time GetEntityTime () const
get current entity time

const Math::bbox & GetLocalBoundingBox ()
get the local space bounding box

const Math::bbox & GetGlobalBoundingBox ()
get bounding box in global space

void ClearLinks (LinkType linkType)
clear all visibility links

void AddLink (LinkType linkType, Ptr< InternalGraphicsEntity > &entity)
add visibility link

const Util::Array< Ptr< InternalGraphicsEntity > > & GetLinks (LinkType type) const
get visibility links by type

virtual Math::ClipStatus::Type ComputeClipStatus (const Math::bbox &box)
compute clip status against bounding box

void MarkRemove ()
mark the entity for removal from the stage at the next possible time

bool IsMarkedForRemove () const
return true if this entity has been marked for removal

void AddDeferredMessage (const Messaging::Message > &msg)
add a message for deferred handling once the object becomes valid

int GetRefCount () const
get the current refcount
```
<table>
<thead>
<tr>
<th>void</th>
<th><strong>AddRef</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>increment refcount by one</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>Release</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>decrement refcount and destroy object if is zero</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>Util::String</strong> &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>OnResolveVisibility</strong> ()</td>
<td>called from internal view</td>
</tr>
<tr>
<td>virtual void <strong>OnTransformChanged</strong> ()</td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td>void <strong>SetLightType</strong> (LightType::Code c)</td>
<td>set the light type (must be called from sub-classes constructor)</td>
</tr>
<tr>
<td>virtual void <strong>OnRenderDebug</strong> ()</td>
<td>called to render a debug visualization of the entity</td>
</tr>
<tr>
<td>void <strong>UpdateShadowTransforms</strong> ()</td>
<td>update shadow transforms</td>
</tr>
<tr>
<td>void <strong>SetSharedData</strong> (const Ptr<a href="">FrameSync::FrameSyncSharedData</a> &amp;data)</td>
<td>set pointer to shared data object</td>
</tr>
<tr>
<td>void <strong>SetType</strong> (InternalGraphicsEntityType::Code t)</td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td>void <strong>SetValid</strong> (bool b)</td>
<td>set to valid state (when the entity becomes ready for rendering)</td>
</tr>
<tr>
<td>void <strong>UpdateClipStatus</strong> (Math::ClipStatus::Type c)</td>
<td>update current clip status</td>
</tr>
<tr>
<td>void <strong>UpdateTime</strong> (Timing::Time time, Timing::Time timeFactor)</td>
<td>update current time</td>
</tr>
<tr>
<td>virtual void <strong>OnActivate</strong> ()</td>
<td>called when entity is created</td>
</tr>
<tr>
<td>virtual void <strong>OnDeactivate</strong> ()</td>
<td>called before entity is destroyed</td>
</tr>
<tr>
<td>virtual void <strong>OnAttachToStage</strong> (const Ptr&lt;InternalStage&gt; &amp;stage)</td>
<td>called when attached to Stage</td>
</tr>
<tr>
<td>virtual void <strong>OnRemoveFromStage</strong> ()</td>
<td>called when removed from Stage</td>
</tr>
<tr>
<td>virtual void <strong>OnSetupSharedData</strong> ()</td>
<td>called to setup the client-portion of the shared data object</td>
</tr>
<tr>
<td>virtual void <strong>OnDiscardSharedData</strong> ()</td>
<td></td>
</tr>
</tbody>
</table>
called to discard the client-portion of the shared data object

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>OnResetSharedData</strong> ()</td>
<td>called per frame to reset the shared data object</td>
</tr>
<tr>
<td>virtual void <strong>OnShow</strong> ()</td>
<td>called when the entity becomes visible</td>
</tr>
<tr>
<td>virtual void <strong>OnHide</strong> ()</td>
<td>called when the entity becomes invisible</td>
</tr>
<tr>
<td>virtual void <strong>OnCullBefore</strong> (Timing::Time time, Timing::Time globalTimeFactor, IndexT frameIndex)</td>
<td>called before culling on each(!) graphics entity (visible or not!)</td>
</tr>
<tr>
<td>virtual void <strong>OnNotifyCullingVisible</strong> (const Ptr&lt;InternalGraphicsEntity&gt; &amp;observer, IndexT frameIndex)</td>
<td>called when the entity has been found visible during culling, may be called several times per frame!</td>
</tr>
<tr>
<td>virtual void <strong>OnRenderBefore</strong> (IndexT frameIndex)</td>
<td>called right before rendering</td>
</tr>
<tr>
<td>void <strong>SetLocalBoundingBox</strong> (const Math::bbox &amp;b)</td>
<td>set the local space bounding box</td>
</tr>
<tr>
<td>void <strong>UpdateGlobalBoundingBox</strong> ()</td>
<td>update the global bounding box from the transform and local box</td>
</tr>
<tr>
<td>void <strong>HandleDeferredMessages</strong> ()</td>
<td>handle deferred messages (called by subclasses once resources are loaded)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Lighting::InternalAbstractLightEntity::HandleMessage(const Ptr<Messaging::Message> &msg) [virtual]
```

handle a message

Handle a message, override this method accordingly in subclasses!

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

Reimplemented in `Lighting::InternalGlobalLightEntity`, `Lighting::InternalPointLightEntity`, and `Lighting::InternalSpotLightEntity`.

```cpp
void Lighting::InternalAbstractLightEntity::OnResolveVisibility() [protected, virtual]
```

called from internalview

This method is called whenever the internalview comes to its Render method. Add light entities to the `LightServer` or to the `ShadowServer`.

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
ClipStatus::Type InternalGraphics::InternalGraphicsEntity::ComputeClipStatus(const Math::bbox box) [virtual, inherited]
```

compute clip status against bounding box

Compute the clip status between this entity and a bounding box in global space. This method must be overwritten in a derived class.

Reimplemented in `InternalGraphics::InternalCameraEntity`, `Lighting::InternalGlobalLightEntity`, `Lighting::InternalPointLightEntity`, and `Lighting::InternalSpotLightEntity`. 
add a message for deferred handling once the object becomes valid.

Message handlers may decide to defer message handling until the object has become valid.

called when entity is created

Activate the entity. This method is called when the entity is created and attached to the graphics server. During OnActivate() the entity should perform any one-time initializations.

Reimplemented in InternalGraphics::InternalModelEntity.

called before entity is destroyed

Deactivate the entity, this method is called when the entity is removed from the graphics server. Any initialization done in OnActivate() should be undone here.

Reimplemented in InternalGraphics::InternalCameraEntity, and InternalGraphics::InternalModelEntity.

called when attached to Stage

This method is called when the graphics entity is attached to a stage. An entity may only be attached to one stage at any time, but can be attached to different stages during its lifetime. Attaching an entity to a stage may be relatively slow because the entity must be inserted into
the cell hierarchy.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnRemoveFromStage() [protected, virtual, inherited]
```
called when removed from Stage

This method is called when the graphics entity is removed from a stage.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnSetupSharedData() [protected, virtual, inherited]
```
called to setup the client-portion of the shared data object

This method is called from **OnActivate()** to setup the shared data object of the entity. The method must call the ClientSetup() method on the sharedData object with the same template type as the main-thread side entity.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnDiscardSharedData() [protected, virtual, inherited]
```
called to discard the client-portion of the shared data object

Called from **OnDeactivate()** to discard the shared data object of the entity from the client side.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnResetSharedData() [protected, virtual, inherited]
```
called per frame to reset the shared data object

This method is called once per frame before **OnCullBefore()** to initialize the shared data object with suitable data (which may be overwritten with up-to-date-data later in the frame). This is necessary because the SharedData object is double buffered, and thus if an update is missed for one frame invalid data from the previous frame
may "leak" into the next frame.

Reimplemented in InternalGraphics::InternalModelEntity.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnShow() [protected, virtual, inherited]
called when the entity becomes visible

This method is called from the SetVisible() method when the entity changes from invisible to visible state.

Reimplemented in InternalGraphics::InternalModelEntity.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnHide() [protected, virtual, inherited]
called when the entity becomes invisible

This method is called from the SetVisible() method when the entity changes from visible to invisible state.

Reimplemented in InternalGraphics::InternalModelEntity.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnCullBefore(Timing::Time time_,
              Timing::Time timeFactor_,
              IndexT frameIndex)
[protected, virtual, inherited]
called before culling on each(!) graphics entity (visible or not!)

This method is called at the beginning of a frame before culling happens on EVERY entity.

Reimplemented in InternalGraphics::InternalModelEntity.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnNotifyCullingVisible(const Ptr<InternalGraphicsEntity> observer &,
            IndexT frameIndex)
```
called when the entity has been found visible during culling, may be called several times per frame!

This method is called during visibility linking when an observed entity is found to be visible by an observer (a camera or a light entity). NOTE that this method will be called several times per frame, so it may be a good idea to use the graphics server's frame counter to protect expensive code from multiple execution.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnRenderBefore(IndexT frameIndex) [protected, virtual, inherited]
```
called right before rendering

This method is called on the entity from **InternalView::Render()** once per frame for every visible entity.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::HandleDeferredMessages() [protected, inherited]
```
handle deferred messages (called by subclasses once resources are loaded)

This method is called once when an object with deferred validation (e.g. ModelEntities) become valid (usually after their resources have finished loading). Any deferred messages will be processed here.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Lighting::InternalGlobalLightEntity
# include <internalgloballightentity.h>

Inheritance diagram for Lighting::InternalGlobalLightEntity:
Detailed Description

Implements a global directional light intended for outdoor areas or as the basis for indoor rendering. Besides the primary directional light component, a global light also usually provides some sort of ambient component for areas in shadow. There should only be one GlobalLight active during rendering a frame. The primary light is directed along the negative z-axis of the light's transform matrix. Scaling and position has no effect on the global light.

(C) 2007 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><strong>LinkType</strong></td>
</tr>
<tr>
<td></td>
<td>visibility link types</td>
</tr>
<tr>
<td>typedef</td>
<td><strong>Id</strong></td>
</tr>
<tr>
<td></td>
<td>a unique id for graphics entities</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InternalGlobalLightEntity</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual Math::ClipStatus::Type <strong>ComputeClipStatus</strong> (const Math::bbox &amp;box)</td>
<td>compute clip status against bounding box</td>
</tr>
<tr>
<td>void <strong>SetBackLightColor</strong> (const Math::float4 &amp;c)</td>
<td>set the backlight color</td>
</tr>
<tr>
<td>const Math::float4 &amp; <strong>GetBackLightColor</strong> () const</td>
<td>get the backlight color</td>
</tr>
<tr>
<td>const Math::vector &amp; <strong>GetLightDirection</strong> () const</td>
<td>get the light direction (extracted from negative z-axis of transformation matrix)</td>
</tr>
<tr>
<td>const Math::float4 &amp; <strong>GetAmbientLightColor</strong> () const</td>
<td>get AmbientLightColor</td>
</tr>
<tr>
<td>void <strong>SetAmbientLightColor</strong> (const Math::float4 &amp;val)</td>
<td>set AmbientLightColor</td>
</tr>
<tr>
<td>const float &amp; <strong>GetBackLightOffset</strong> () const</td>
<td>get BackLightOffset</td>
</tr>
<tr>
<td>void <strong>SetBackLightOffset</strong> (const float &amp;val)</td>
<td>set BackLightOffset</td>
</tr>
<tr>
<td>virtual void <strong>HandleMessage</strong> (const Ptr<a href="">Messaging::Message</a> &amp;msg)</td>
<td>handle a message</td>
</tr>
<tr>
<td>LightType::Code <strong>GetLightType</strong> () const</td>
<td>get the light type</td>
</tr>
<tr>
<td>void <strong>SetColor</strong> (const Math::float4 &amp;c)</td>
<td>set primary light color</td>
</tr>
<tr>
<td>const Math::float4 &amp; <strong>GetColor</strong> () const</td>
<td>get primary light color</td>
</tr>
<tr>
<td>void <strong>SetCastShadows</strong> (bool b)</td>
<td>enable/disable shadow casting</td>
</tr>
<tr>
<td>bool</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetCastShadows</code> () const</td>
<td>get shadow casting flag</td>
</tr>
<tr>
<td><code>void SetProjMapUvOffsetAndScale</code> (const <code>Math::float4</code> &amp;v)</td>
<td>set projection map UV offset and scale (xyz-&gt;offset, zw-&gt;scale)</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetProjMapUvOffsetAndScale</code> const</td>
<td>get projection map UV offset and scale</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetInvTransform</code> () const</td>
<td>get inverse transform (transforms from world space to light space)</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetProjTransform</code> () const</td>
<td>get light-projection matrix (transforms from world space to light projection space)</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetInvLightProjTransform</code> ()</td>
<td>get world-to-light-projection transform (transforms from world to light projection space)</td>
</tr>
<tr>
<td><code>void SetShadowBufferUvOffsetAndScale</code> (const <code>Math::float4</code> &amp;uvOffset)</td>
<td>set shadow buffer uv rectangle (optionally set by light/shadow servers)</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetShadowBufferUvOffsetAndScale</code> () const</td>
<td>get shadow buffer uv rectangle</td>
</tr>
<tr>
<td><code>bool GetCastShadowsThisFrame</code></td>
<td>get cast shadows this frame</td>
</tr>
<tr>
<td><code>void SetCastShadowsThisFrame</code> (val)</td>
<td>set CastShadowsThisFrame</td>
</tr>
<tr>
<td><code>void SetShadowTransform</code> (const <code>Math::matrix44</code> &amp;val)</td>
<td>set shadow transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetShadowInvTransform</code> ()</td>
<td>get shadow transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetShadowInvLightProjTransform</code> ()</td>
<td>get shadow projection transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetShadowProjTransform</code> ()</td>
<td>get shadow projection transform</td>
</tr>
</tbody>
</table>
#### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>float GetShadowIntensity()</code></td>
<td>get ShadowIntensity</td>
</tr>
<tr>
<td><code>void SetShadowIntensity(float val)</code></td>
<td>set ShadowIntensity</td>
</tr>
<tr>
<td><code>bool IsActive()</code></td>
<td>return true if entity is currently active (is in OnActivate()/OnDeactivate())</td>
</tr>
<tr>
<td><code>bool IsValid()</code></td>
<td>return true if entity is current valid (ready for rendering)</td>
</tr>
<tr>
<td><code>Id GetId()</code></td>
<td>get the graphics entity's unique id</td>
</tr>
<tr>
<td><code>InternalGraphicsEntityType::Code GetType()</code></td>
<td>get the entity type</td>
</tr>
<tr>
<td><code>void SetTransform(const Math::matrix44 &amp;m)</code></td>
<td>set the entity's world space transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetTransform()</code></td>
<td>get the entity's world space transform</td>
</tr>
<tr>
<td><code>void SetVisible(bool b)</code></td>
<td>set the entity's visibility</td>
</tr>
<tr>
<td><code>bool IsVisible()</code></td>
<td>return true if entity is set to visible</td>
</tr>
<tr>
<td><code>const Ptr&lt; InternalStage &gt; &amp; GetStage()</code></td>
<td>get the stage this entity is attached to</td>
</tr>
<tr>
<td><code>bool IsAttachedToStage()</code></td>
<td>return true if entity is attached to stage</td>
</tr>
<tr>
<td><code>Timing::Time GetEntityTime()</code></td>
<td>get current entity time</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetLocalBoundingBox()</code></td>
<td>get the local space bounding box</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetGlobalBoundingBox()</code></td>
<td>get bounding box in global space</td>
</tr>
<tr>
<td><code>void ClearLinks(LinkType linkType)</code></td>
<td>clear all visibility links</td>
</tr>
<tr>
<td><code>void AddLink(LinkType linkType, Ptr&lt; InternalGraphicsEntity &gt;)</code></td>
<td>add a visibility link</td>
</tr>
</tbody>
</table>
add visibility link

const `Util::Array<Ptr<InternalGraphicsEntity>> &entity)`

`GetLinks (LinkType type) const` get visibility links by type

void `MarkRemove ()` mark the entity for removal from the stage next possible time

bool `IsMarkedForRemove () const` return true if this entity has been marked removal

void `AddDeferredMessage (const Messaging::Message &msg)` add a message for deferred handling once object becomes valid

int `GetRefCount () const` get the current refcount

void `AddRef ()` increment refcount by one

void `Release ()` decrement refcount and destroy object if is zero

bool `IsInstanceOf (const Rtti &rtti)` return true if this object is instance of given class

bool `IsInstanceOf (const Util::String &className)` const return true if this object is instance of given class by string

bool `IsInstanceOf (const Util::FourCC &classFourCC)` const return true if this object is instance of given class by fourcc

bool `IsA (const Rtti &rtti) const` return true if this object is instance of given or a derived class

bool `IsA (const Util::String &rttiName)` const return true if this object is instance of given or a derived class, by string

bool `IsA (const Util::FourCC &rttiFourCC)` const return true if this object is instance of given or a derived class, by fourcc
```plaintext
const
return true if this object is instance of given or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
```
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <code>OnTransformChanged()</code></td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td>virtual void <code>OnRenderDebug()</code></td>
<td>called to render a debug visualization of the entity</td>
</tr>
<tr>
<td>virtual void <code>OnResolveVisibility()</code></td>
<td>called from <code>internalview</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>SetLightType</code>(LightType::Code c)</td>
<td>set the light type (must be called from sub-classes constructor</td>
</tr>
<tr>
<td>void <code>UpdateShadowTransforms</code></td>
<td>update shadow transforms</td>
</tr>
<tr>
<td>void <code>SetSharedData</code>(const Ptr&lt; FrameSync::FrameSyncSharedData &gt; &amp;data)</td>
<td>set pointer to shared data object</td>
</tr>
<tr>
<td>void <code>setType</code>(InternalGraphicsEntityType::Code t)</td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td>void <code>SetValid</code>(bool b)</td>
<td>set to valid state (when the entity becomes ready for rendering)</td>
</tr>
<tr>
<td>void <code>UpdateClipStatus</code>(Math::ClipStatus::Type c)</td>
<td>update current clip status</td>
</tr>
<tr>
<td>void <code>UpdateTime</code>(Timing::Time time, Timing::Time timeFactor)</td>
<td>update current time</td>
</tr>
<tr>
<td>virtual void <code>OnActivate</code>()</td>
<td>called when entity is created</td>
</tr>
<tr>
<td>virtual void <code>OnDeactivate</code>()</td>
<td>called before entity is destroyed</td>
</tr>
<tr>
<td>virtual void <code>OnAttachToStage</code>(const Ptr&lt; InternalStage &gt; &amp;stage)</td>
<td>called when attached to <code>Stage</code></td>
</tr>
<tr>
<td>virtual void <code>OnRemoveFromStage</code>()</td>
<td>called when removed from <code>Stage</code></td>
</tr>
<tr>
<td>virtual void <code>OnSetupSharedData</code>()</td>
<td>called to setup the client-portion of the shared data object</td>
</tr>
<tr>
<td>virtual void <code>OnDiscardSharedData</code>()</td>
<td>called when the shared data is discarded</td>
</tr>
</tbody>
</table>
called to discard the client-portion of the shared data object

**virtual void** **OnResetSharedData** ()  
called per frame to reset the shared data object

**virtual void** **OnShow** ()  
called when the entity becomes visible

**virtual void** **OnHide** ()  
called when the entity becomes invisible

**virtual void** **OnCullBefore** (**Timing::Time** time, **Timing::Time** globalTimeFactor, IndexT frameIndex)  
called before culling on each(!) graphics entity (visible or not!)

**virtual void** **OnNotifyCullingVisible** (const **Ptr**< **InternalGraphicsEntity** >&observer, IndexT frameIndex)  
called when the entity has been found visible during culling, may be called several times per frame!

**virtual void** **OnRenderBefore** (IndexT frameIndex)  
called right before rendering

**void** **SetLocalBoundingBox** (const **Math::bbox** &b)  
set the local space bounding box

**void** **UpdateGlobalBoundingBox** ()  
update the global bounding box from the transform and local box

**void** **HandleDeferredMessages** ()  
handle deferred messages (called by subclasses once resources are loaded)
Member Function Documentation

```cpp
void Lighting::InternalGlobalLightEntity::HandleMessage(const Ptr<Message> msg) [virtual]
```

handle a message

Handle a message, override this method accordingly in subclasses!

Reimplemented from `Lighting::InternalAbstractLightEntity`.

```cpp
void Lighting::InternalAbstractLightEntity::OnResolveVisibility()
```

called from internalview

This method is called whenever the internalview comes to its Render method. Add light entities to the `LightServer` or to the `ShadowServer`.

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::AddDeferredMessage(const Ptr<Message> msg)
```

add a message for deferred handling once the object becomes valid

Message handlers may decide to defer message handling until the object has become valid.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnActivate()
```

called when entity is created

Activate the entity. This method is called when the entity is created an attached to the graphics server. During `OnActivate()` the entity should perform any one-time initializations.
Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnDeactivate( ) [protected, virtual, inherited]
```
called before entity is destroyed

Deactivate the entity, this method is called when the entity is removed from the graphics server. Any initialization done in `OnActivate()` should be undone here.

Reimplemented in `InternalGraphics::InternalCameraEntity`, and `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnAttachToStage( const Ptr< InternalStage > & s ) [protected, virtual, inherited]
```
called when attached to Stage

This method is called when the graphics entity is attached to a stage. An entity may only be attached to one stage at any time, but can be attached to different stages during its lifetime. Attaching an entity to a stage may be relatively slow because the entity must be inserted into the cell hierarchy.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnRemoveFromStage( ) [protected, virtual, inherited]
```
called when removed from Stage

This method is called when the graphics entity is removed from a stage.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnSetupSharedData( ) [protected, virtual, inherited]
```
called to setup the client-portion of the shared data object

This method is called from `OnActivate()` to setup the shared data object of the entity. The method must call the `ClientSetup()` method on the `sharedData` object with the same template type as the main-thread.
side entity.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnDiscardSharedData( ) [protected, virtual, inherited]
```
called to discard the client-portion of the shared data object

Called from **OnDeactivate()** to discard the shared data object of the entity from the client side.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnResetSharedData( ) [protected, virtual, inherited]
```
called per frame to reset the shared data object

This method is called once per frame before **OnCullBefore()** to initialize the shared data object with suitable data (which may be overwritten with up to date data later in the frame). This is necessary because the SharedData object is double buffered, and thus if an update if missed for one frame invalid data from the previous frame may "leak" into the next frame.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnShow( ) [protected, virtual, inherited]
```
called when the entity becomes visible

This method is called from the **SetVisible()** method when the entity changes from invisible to visible state.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnHide( ) [protected, virtual, inherited]
```
called when the entity becomes invisible
This method is called from the `SetVisible()` method when the entity changes from visible to invisible state.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnCullBefore(
    Timing::Time time_,
   Timing::Time timeFactor_,
    IndexT frameIndex
) [protected, virtual, inherited]
```
called before culling on each(!) graphics entity (visible or not!)

This method is called at the beginning of a frame before culling happens on EVERY entity.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnNotifyCullingVisible(const Ptr<InternalGraphicsEntity> &observer, 
    IndexT frameIndex)
```
called when the entity has been found visible during culling, may be called several times per frame!

This method is called during visibility linking when an observed entity is found to be visible by an observer (a camera or a light entity). NOTE that this method will be called several times per frame, so it may be a good idea to use the graphics server's frame counter to protect expensive code from multiple execution.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnRenderBefore (IndexT frameIndex) [protected, virtual, inherited]
```
called right before rendering
This method is called on the entity from `InternalView::Render()` once per frame for every visible entity.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::HandleDeferredMessages()
```[protected, inherited]

handle deferred messages (called by subclasses once resources are loaded)

This method is called once when an object with deferred validation (e.g. ModelEntities) become valid (usually after their resources have finished loading). Any deferred messages will be processed here.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.
Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Lighting::InternalPointLightEntity
Lighting::InternalPointLightEntity Class Reference

#include <internalpointlightentity.h>

Inheritance diagram for Lighting::InternalPointLightEntity:
Detailed Description

Implements a local spot light. The spot light's cone is computed from the transformation matrix of the light entity. The spot light cone will point along -z, and the cone's angle will be defined by the length of x and y component at the end of the z component.

(C) 2007 Radon Labs GmbH
# Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>LinkType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>visibility link types</td>
</tr>
<tr>
<td>typedef</td>
<td>Id</td>
</tr>
<tr>
<td>IndexT</td>
<td>a unique id for graphics entities</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function/Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InternalPointLightEntity</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual Math::ClipStatus::Type ComputeClipStatus</strong> (const Math::bbox &amp;box)</td>
<td>compute clip status against bounding box</td>
</tr>
<tr>
<td><strong>virtual void HandleMessage</strong> (const Ptr<a href="">Messaging::Message</a> &amp;msg)</td>
<td>handle a message</td>
</tr>
<tr>
<td><strong>LightType::Code GetLightType</strong> () const</td>
<td>get the light type</td>
</tr>
<tr>
<td><strong>void SetColor</strong> (const Math::float4 &amp;c)</td>
<td>set primary light color</td>
</tr>
<tr>
<td><strong>const Math::float4 &amp; GetColor</strong> () const</td>
<td>get primary light color</td>
</tr>
<tr>
<td><strong>void SetCastShadows</strong> (bool b)</td>
<td>enable/disable shadow casting</td>
</tr>
<tr>
<td><strong>bool GetCastShadows</strong> () const</td>
<td>get shadow casting flag</td>
</tr>
<tr>
<td><strong>void SetProjMapUvOffsetAndScale</strong> (const Math::float4 &amp;v)</td>
<td>set projection map UV offset and scale (xy-&gt;offset, zw-&gt;scale)</td>
</tr>
<tr>
<td><strong>const Math::float4 &amp; GetProjMapUvOffsetAndScale</strong> const</td>
<td>get projection map UV offset and scale</td>
</tr>
<tr>
<td><strong>const Math::matrix44 &amp; GetInvTransform</strong> () const</td>
<td>get inverse transform (transforms from world to light space)</td>
</tr>
<tr>
<td><strong>const Math::matrix44 &amp; GetProjTransform</strong> () const</td>
<td>get light-projection matrix (transforms from light space to light projection space)</td>
</tr>
<tr>
<td><strong>const Math::matrix44 &amp; GetInvLightProjTransform</strong> () const</td>
<td>get world-to-light-projection transform (transforms from world to light projection space)</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void SetShadowBufferUvOffsetAndScale</code></td>
<td>Set shadow buffer uv rectangle (optionally set by light/shadow servers)</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; uvOffset</code></td>
<td></td>
</tr>
<tr>
<td><code>GetShadowBufferUvOffsetAndScale()</code></td>
<td>Get shadow buffer uv rectangle</td>
</tr>
<tr>
<td><code>bool GetCastShadowsThisFrame</code></td>
<td>Get cast shadows this frame</td>
</tr>
<tr>
<td><code>void SetCastShadowsThisFrame (val)</code></td>
<td>Set CastShadowsThisFrame</td>
</tr>
<tr>
<td><code>void SetShadowTransform</code> (const Math::matrix44 &amp; val)`</td>
<td>Set shadow transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>GetShadowInvTransform()</code></td>
<td>Get shadow transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>GetShadowInvLightProjTransform</code></td>
<td>Get shadow projection transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>GetShadowProjTransform()</code></td>
<td>Get shadow projection transform</td>
</tr>
<tr>
<td><code>float GetShadowIntensity()</code></td>
<td>Get ShadowIntensity</td>
</tr>
<tr>
<td><code>void SetShadowIntensity (float val)</code></td>
<td>Set ShadowIntensity</td>
</tr>
<tr>
<td><code>bool IsActive ()</code></td>
<td>Return true if entity is currently active (is between OnActivate()/OnDeactivate())</td>
</tr>
<tr>
<td><code>bool IsValid ()</code></td>
<td>Return true if entity is current valid (ready rendering)</td>
</tr>
<tr>
<td><code>Id GetId ()</code></td>
<td>Get the graphics entity's unique id</td>
</tr>
<tr>
<td><code>InternalGraphicsEntityType::Code</code></td>
<td></td>
</tr>
<tr>
<td><code>GetType ()</code></td>
<td>Get the entity type</td>
</tr>
<tr>
<td><code>void SetTransform</code> (const Math::matrix44 &amp; m)`</td>
<td>Set the entity's world space transform</td>
</tr>
</tbody>
</table>
const Math::matrix44 & GetTransform() const
get the entity's world space transform

void SetVisible(bool b)
set the entity's visibility

bool IsVisible() const
return true if entity is set to visible

const Ptr< InternalStage > & GetStage() const
get the stage this entity is attached to

bool IsAttachedToStage() const
return true if entity is attached to stage

Timing::Time GetEntityTime() const
get current entity time

const Math::bbox & GetLocalBoundingBox() const
get the local space bounding box

const Math::bbox & GetGlobalBoundingBox() const
get bounding box in global space

void ClearLinks(LinkType linkType)
clear all visibility links

void AddLink(LinkType linkType, Ptr< InternalGraphicsEntity > &entity)
add visibility link

const Util::Array< Ptr< InternalGraphicsEntity > > & GetLinks(LinkType type) const
get visibility links by type

void MarkRemove()
mark the entity for removal from the stage at the next possible time

bool IsMarkedForRemove() const
return true if this entity has been marked for removal

void AddDeferredMessage(const Messaging::Message > &msg)
add a message for deferred handling once the object becomes valid

int GetRefCount() const
get the current refcount

void AddRef()
increment refcount by one
<table>
<thead>
<tr>
<th>void</th>
<th>Release ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>decrement refcount and destroy object if refcount is zero</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::String &amp;className) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class by string</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class by fourcc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class or a derived class</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class or a derived class, by string</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class or a derived class, by fourcc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class name</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
</table>

*dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)*
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <code>OnTransformChanged</code> ()</td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td>virtual void <code>OnRenderDebug</code> ()</td>
<td>called to render a debug visualization of the entity</td>
</tr>
<tr>
<td>virtual void <code>OnResolveVisibility</code> ()</td>
<td>called from internalview</td>
</tr>
<tr>
<td>void <code>SetLightType</code> (LightType::Code c)</td>
<td>set the light type (must be called from sub-classes constructor)</td>
</tr>
<tr>
<td>void <code>UpdateShadowTransforms</code> ()</td>
<td>update shadow transforms</td>
</tr>
<tr>
<td>void <code>SetSharedData</code> (const Ptr&lt; FrameSync::FrameSyncSharedData &gt; &amp;data)</td>
<td>set pointer to shared data object</td>
</tr>
<tr>
<td>void <code>setType</code> (InternalGraphicsEntityType::Code t)</td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td>void <code>SetValid</code> (bool b)</td>
<td>set to valid state (when the entity becomes ready for rendering)</td>
</tr>
<tr>
<td>void <code>UpdateClipStatus</code> (Math::ClipStatus::Type c)</td>
<td>update current clip status</td>
</tr>
<tr>
<td>void <code>UpdateTime</code> (Timing::Time time, Timing::Time timeFactor)</td>
<td>update current time</td>
</tr>
<tr>
<td>virtual void <code>OnActivate</code> ()</td>
<td>called when entity is created</td>
</tr>
<tr>
<td>virtual void <code>OnDeactivate</code> ()</td>
<td>called before entity is destroyed</td>
</tr>
<tr>
<td>virtual void <code>OnAttachToStage</code> (const Ptr&lt; InternalStage &gt; &amp;stage)</td>
<td>called when attached to Stage</td>
</tr>
<tr>
<td>virtual void <code>OnRemoveFromStage</code> ()</td>
<td>called when removed from Stage</td>
</tr>
<tr>
<td>virtual void <code>OnSetupSharedData</code> ()</td>
<td>called to setup the client-portion of the shared data object</td>
</tr>
<tr>
<td>virtual void <code>OnDiscardSharedData</code> ()</td>
<td></td>
</tr>
</tbody>
</table>
called to discard the client-portion of the shared data object

virtual void **OnResetSharedData** ()
called per frame to reset the shared data object

virtual void **OnShow** ()
called when the entity becomes visible

virtual void **OnHide** ()
called when the entity becomes invisible

virtual void **OnCullBefore** (Timing::Time time, Timing::Time globalTimeFactor, IndexT frameIndex)
called before culling on each(!) graphics entity (visible or not!)

virtual void **OnNotifyCullingVisible** (const Ptr<InternalGraphicsEntity> &observer, IndexT frameIndex)
called when the entity has been found visible during culling, may be called several times per frame!

virtual void **OnRenderBefore** (IndexT frameIndex)
called right before rendering

void **SetLocalBoundingBox** (const Math::bbox &b)
set the local space bounding box

void **UpdateGlobalBoundingBox** ()
update the global bounding box from the transform and local box

void **HandleDeferredMessages** ()
handle deferred messages (called by subclasses once resources are loaded)
Member Function Documentation

```cpp
void Lighting::InternalPointLightEntity::HandleMessage (const Messaging::Message msg) [virtual]
handle a message
Handle a message, override this method accordingly in subclasses!
Reimplemented from Lighting::InternalAbstractLightEntity.
```

```cpp
void Lighting::InternalAbstractLightEntity::OnResolveVisibility () [protected, virtual, inherited]
called from internalview
This method is called whenever the the internalview comes to its Render method. Add light entities to the LightServer or to the ShadowServer.
Reimplemented from InternalGraphics::InternalGraphicsEntity.
```

```cpp
void InternalGraphics::InternalGraphicsEntity::AddDeferredMessage (const Messaging::Message msg) []::
add a message for deferred handling once the object becomes valid
Message handlers may decide to defer message handling until the object has become valid.
```

```cpp
void InternalGraphics::InternalGraphicsEntity::OnActivate () [protected, virtual, inherited]
called when entity is created
Activate the entity. This method is called when the entity is created an attached to the graphics server. During OnActivate() the entity should perform any one-time initializations.
Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnDeactivate() [protected, virtual, inherited]
```
called before entity is destroyed

Deactivate the entity, this method is called when the entity is removed from the graphics server. Any initialization done in `OnActivate()` should be undone here.

Reimplemented in `InternalGraphics::InternalCameraEntity`, and `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnAttachToStage(const Ptr<InternalStage> & s) [protected, virtual, inherited]
```
called when attached to Stage

This method is called when the graphics entity is attached to a stage. An entity may only be attached to one stage at any time, but can be attached to different stages during its lifetime. Attaching an entity to a stage may be relatively slow because the entity must be inserted into the cell hierarchy.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnRemoveFromStage() [protected, virtual, inherited]
```
called when removed from Stage

This method is called when the graphics entity is removed from a stage.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnSetupSharedData() [protected, virtual, inherited]
```
called to setup the client-portion of the shared data object

This method is called from `OnActivate()` to setup the shared data object of the entity. The method must call the ClientSetup() method on the sharedData object with the same template type as the main-thread
side entity.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnDiscardSharedData() [protected, virtual, inherited]
```
called to discard the client-portion of the shared data object

Called from `OnDeactivate()` to discard the shared data object of the entity from the client side.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnResetSharedData() [protected, virtual, inherited]
```
called per frame to reset the shared data object

This method is called once per frame before `OnCullBefore()` to initialize the shared data object with suitable data (which may be overwritten with up to date data later in the frame). This is necessary because the SharedData object is double buffered, and thus if an update is missed for one frame invalid data from the previous frame may "leak" into the next frame.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnShow() [protected, virtual, inherited]
```
called when the entity becomes visible

This method is called from the `SetVisible()` method when the entity changes from invisible to visible state.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnHide() [protected, virtual, inherited]
```
called when the entity becomes invisible
This method is called from the `SetVisible()` method when the entity changes from visible to invisible state.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```c++
void InternalGraphics::InternalGraphicsEntity::OnCullBefore ( Timing::Time time_,
    Timing::Time timeFactor_,
    IndexT frameIndex )
```

called before culling on each(!) graphics entity (visible or not!)

This method is called at the beginning of a frame before culling happens on EVERY entity.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```c++
void InternalGraphics::InternalGraphicsEntity::OnNotifyCullingVisible ( const Ptr< InternalGraphicsEntity >& observer,
    IndexT frameIndex )
```

called when the entity has been found visible during culling, may be called several times per frame!

This method is called during visibility linking when an observed entity is found to be visible by an observer (a camera or a light entity). NOTE that this method will be called several times per frame, so it may be a good idea to use the graphics server's frame counter to protect expensive code from multiple execution.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```c++
void InternalGraphics::InternalGraphicsEntity::OnRenderBefore ( IndexT frameIndex )
```

called right before rendering
This method is called on the entity from `InternalView::Render()` once per frame for every visible entity.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::HandleDeferredMessages() [protected, inherited]
```

handle deferred messages (called by subclasses once resources are loaded)

This method is called once when an object with deferred validation (e.g. ModelEntities) become valid (usually after their resources have finished loading). Any deferred messages will be processed here.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.
Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Lighting::InternalSpotLightEntity
Lighting::InternalSpotLightEntity Class Reference

#include <internalspotlightentity.h>

Inheritance diagram for Lighting::InternalSpotLightEntity:
Detailed Description

Implements a local spot light. The spot light's cone is computed from the transformation matrix of the light entity. The spot light cone will point along -z, and the cone's angle will be defined by the length of x and y component at the end of the z component.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><code>LinkType</code></td>
</tr>
<tr>
<td></td>
<td>visibility link types</td>
</tr>
<tr>
<td>typedef</td>
<td><code>Id</code></td>
</tr>
<tr>
<td></td>
<td><em>a unique id for graphics entities</em></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InternalSpotLightEntity()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>virtual Math::ClipStatus::Type ComputeClipStatus(const Math::bbox &amp;box)</code></td>
<td>Compute clip status against bounding box</td>
</tr>
<tr>
<td><code>virtual void HandleMessage(const Ptr&lt;Messaging::Message&gt; &amp;msg)</code></td>
<td>Handle a message</td>
</tr>
<tr>
<td><code>LightType::Code GetLightType()</code></td>
<td>Get the light type</td>
</tr>
<tr>
<td><code>void SetColor(const Math::float4 &amp;c)</code></td>
<td>Set primary light color</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetColor()</code></td>
<td>Get primary light color</td>
</tr>
<tr>
<td><code>void SetCastShadows(bool b)</code></td>
<td>Enable/disable shadow casting</td>
</tr>
<tr>
<td><code>bool GetCastShadows()</code></td>
<td>Get shadow casting flag</td>
</tr>
<tr>
<td><code>void SetProjMapUvOffsetAndScale(const Math::float4 &amp;v)</code></td>
<td>Set projection map UV offset and scale (xy-&gt;offset, zw-&gt;scale)</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetProjMapUvOffsetAndScale()</code></td>
<td>Get projection map UV offset and scale</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetInvTransform()</code></td>
<td>Get inverse transform (transforms from world to light space)</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetProjTransform()</code></td>
<td>Get light-projection matrix (transforms from light space to light projection space)</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetInvLightProjTransform()</code></td>
<td>Get world-to-light-projection transform (transforms from world to light projection space)</td>
</tr>
</tbody>
</table>
void SetShadowBufferUvOffsetAndScale (const Math::float4 &uvOffset)  
set shadow buffer uv rectangle (optionally set by light/shadow servers)

const Math::float4 & GetShadowBufferUvOffsetAndScale () const  
get shadow buffer uv rectangle

bool GetCastShadowsThisFrame  
get cast shadows this frame

void SetCastShadowsThisFrame (bool val)  
set CastShadowsThisFrame

void SetShadowTransform (const Math::matrix44 &val)  
set shadow transform

const Math::matrix44 & GetShadowInvTransform ()  
get shadow transform

const Math::matrix44 & GetShadowInvLightProjTransform ()  
get shadow projection transform

const Math::matrix44 & GetShadowProjTransform ()  
get shadow projection transform

float GetShadowIntensity () const  
get ShadowIntensity

void SetShadowIntensity (float val)  
set ShadowIntensity

bool IsActive () const  
return true if entity is currently active (is between OnActivate() / OnDeactivate())

bool IsValid () const  
return true if entity is current valid (ready rendering)

Id GetId () const  
get the graphics entity's unique id

InternalGraphicsEntityType::Code GetType () const  
get the entity type

void SetTransform (const Math::matrix44 &m)  
set the entity's world space transform
const Math::matrix44 & GetTransform () const
get the entity's world space transform

void SetVisible (bool b)
set the entity's visibility

bool IsVisible () const
return true if entity is set to visible

const Ptr< InternalStage > & GetStage () const
get the stage this entity is attached to

bool IsAttachedToStage () const
return true if entity is attached to stage

Timing::Time GetEntityTime () const
get current entity time

const Math::bbox & GetLocalBoundingBox ()
get the local space bounding box

const Math::bbox & GetGlobalBoundingBox ()
get bounding box in global space

void ClearLinks (LinkType linkType)
clear all visibility links

void AddLink (LinkType linkType, 
Ptr< InternalGraphicsEntity > &entity)
add visibility link

const Util::Array< Ptr< InternalGraphicsEntity > > & 
GetLinks (LinkType type) const
get visibility links by type

void MarkRemove ()
mark the entity for removal from the stage at the next possible time

bool IsMarkedForRemove () const
return true if this entity has been marked for removal

void AddDeferredMessage (const Messaging::Message > &msg)
add a message for deferred handling once the object becomes valid

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one
<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Rtti &amp;rtti)</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>const Util::FourCC</td>
<td>GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <code>OnTransformChanged()</code></td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td>virtual void <code>OnRenderDebug()</code></td>
<td>called to render a debug visualization of the entity</td>
</tr>
<tr>
<td>virtual void <code>OnResolveVisibility()</code></td>
<td>called from <code>internalview</code></td>
</tr>
<tr>
<td>void <code>SetLightType(LightType::Code c)</code></td>
<td>set the light type (must be called from sub-classes constructor)</td>
</tr>
<tr>
<td>void <code>UpdateShadowTransforms()</code></td>
<td>update shadow transforms</td>
</tr>
<tr>
<td>void <code>SetSharedData(const Ptr&lt;FrameSync::FrameSyncSharedData&gt; &amp;data)</code></td>
<td>set pointer to shared data object</td>
</tr>
<tr>
<td>void <code>SetType(InternalGraphicsEntityType::Code t)</code></td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td>void <code>SetValid(bool b)</code></td>
<td>set to valid state (when the entity becomes ready for rendering)</td>
</tr>
<tr>
<td>void <code>UpdateClipStatus(Math::ClipStatus::Type c)</code></td>
<td>update current clip status</td>
</tr>
<tr>
<td>void <code>UpdateTime(Timing::Time time, Timing::Time timeFactor)</code></td>
<td>update current time</td>
</tr>
<tr>
<td>virtual void <code>OnActivate()</code></td>
<td>called when entity is created</td>
</tr>
<tr>
<td>virtual void <code>OnDeactivate()</code></td>
<td>called before entity is destroyed</td>
</tr>
<tr>
<td>virtual void <code>OnAttachToStage(const Ptr&lt;InternalStage&gt; &amp;stage)</code></td>
<td>called when attached to Stage</td>
</tr>
<tr>
<td>virtual void <code>OnRemoveFromStage()</code></td>
<td>called when removed from Stage</td>
</tr>
<tr>
<td>virtual void <code>OnSetupSharedData()</code></td>
<td>called to setup the client-portion of the shared data object</td>
</tr>
<tr>
<td>virtual void <code>OnDiscardSharedData()</code></td>
<td></td>
</tr>
</tbody>
</table>
called to discard the client-portion of the shared data object

**virtual void** `OnResetSharedData ()`
*called per frame to reset the shared data object*

**virtual void** `OnShow ()`
*called when the entity becomes visible*

**virtual void** `OnHide ()`
*called when the entity becomes invisible*

**virtual void** `OnCullBefore (Timing::Time time, Timing::Time globalTimeFactor, IndexT frameIndex)`
*called before culling on each(!) graphics entity (visible or not!)*

**virtual void** `OnNotifyCullingVisible (const Ptr<InternalGraphicsEntity> &observer, IndexT frameIndex)`
*called when the entity has been found visible during culling, may be called several times per frame!*

**virtual void** `OnRenderBefore (IndexT frameIndex)`
*called right before rendering*

**void** `SetLocalBoundingBox (const Math::bbox &b)`
*set the local space bounding box*

**void** `UpdateGlobalBoundingBox ()`
*update the global bounding box from the transform and local box*

**void** `HandleDeferredMessages ()`
*handle deferred messages (called by subclasses once resources are loaded)*
Member Function Documentation

```c++
void Lighting::InternalSpotLightEntity::HandleMessage(const Ptr<Messaging::Message> &msg) [virtual]
```

handle a message

Handle a message, override this method accordingly in subclasses!

Reimplemented from `Lighting::InternalAbstractLightEntity`.

```c++
void Lighting::InternalAbstractLightEntity::OnResolveVisibility() [protected, virtual, inherited]
```

called from internalview

This method is called whenever the the internalview comes to its Render method. Add light entities to the `LightServer` or to the `ShadowServer`.

Reimplemented from `InternalGraphics::InternalGraphicsEntity`.

```c++
void InternalGraphics::InternalGraphicsEntity::AddDeferredMessage(const Ptr<Messaging::Message> &msg) [:]
```

add a message for deferred handling once the object becomes valid

Message handlers may decide to defer message handling until the object has become valid.

```c++
void InternalGraphics::InternalGraphicsEntity::OnActivate() [protected, virtual, inherited]
```

called when entity is created

Activate the entity. This method is called when the entity is created an attached to the graphics server. During `OnActivate()` the entity should perform any one-time initializations.
Reimplemented in \texttt{InternalGraphics::InternalModelEntity}.

\begin{verbatim}
void InternalGraphics::InternalGraphicsEntity::OnDeactivate() [protected, virtual, inherited]

called before entity is destroyed

Deactivate the entity, this method is called when the entity is removed from the graphics server. Any initialization done in \texttt{OnActivate()} should be undone here.

Reimplemented in \texttt{InternalGraphics::InternalCameraEntity}, and \texttt{InternalGraphics::InternalModelEntity}.

\begin{verbatim}
void InternalGraphics::InternalGraphicsEntity::OnAttachToStage(const \texttt{Ptr<InternalStage} &s) [protected, virtual, inherited]

called when attached to Stage

This method is called when the graphics entity is attached to a stage. An entity may only be attached to one stage at any time, but can be attached to different stages during its lifetime. Attaching an entity to a stage may be relatively slow because the entity must be inserted into the cell hierarchy.

\begin{verbatim}
void InternalGraphics::InternalGraphicsEntity::OnRemoveFromStage() [protected, virtual, inherited]

called when removed from Stage

This method is called when the graphics entity is removed from a stage.

\begin{verbatim}
void InternalGraphics::InternalGraphicsEntity::OnSetupSharedData() [protected, virtual, inherited]

called to setup the client-portion of the shared data object

This method is called from \texttt{OnActivate()} to setup the shared data object of the entity. The method must call the \texttt{ClientSetup()} method on the \texttt{sharedData} object with the same template type as the main-thread
side entity.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnDiscardSharedData() [protected, virtual, inherited]
```
called to discard the client-portion of the shared data object

Called from **OnDeactivate()** to discard the shared data object of the entity from the client side.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnResetSharedData() [protected, virtual, inherited]
```
called per frame to reset the shared data object

This method is called once per frame before **OnCullBefore()** to initialize the shared data object with suitable data (which may be overwritten with up-to-date data later in the frame). This is necessary because the SharedData object is double buffered, and thus if an update is missed for one frame invalid data from the previous frame may "leak" into the next frame.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnShow() [protected, virtual, inherited]
```
called when the entity becomes visible

This method is called from the **SetVisible()** method when the entity changes from invisible to visible state.

Reimplemented in **InternalGraphics::InternalModelEntity**.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnHide() [protected, virtual, inherited]
```
called when the entity becomes invisible
This method is called from the `SetVisible()` method when the entity changes from visible to invisible state.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnCullBefore(
    Timing::Time time_,
    Timing::Time timeFactor_,
    IndexT frameIndex
) [protected, virtual, inherited]
```
called before culling on each(!) graphics entity (visible or not!)

This method is called at the beginning of a frame before culling happens on EVERY entity.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnNotifyCullingVisible(
    const Ptr<InternalGraphicsEntity> &observer,
    IndexT frameIndex
)
```
called when the entity has been found visible during culling, may be called several times per frame!

This method is called during visibility linking when an observed entity is found to be visible by an observer (a camera or a light entity). NOTE that this method will be called several times per frame, so it may be a good idea to use the graphics server's frame counter to protect expensive code from multiple execution.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::OnRenderBefore(
    IndexT frameIndex
) [protected, virtual, inherited]
```
called right before rendering
This method is called on the entity from `InternalView::Render()` once per frame for every visible entity.

Reimplemented in `InternalGraphics::InternalModelEntity`.

```cpp
void InternalGraphics::InternalGraphicsEntity::HandleDeferredMessages() [protected, inherited]
```

handle deferred messages (called by subclasses once resources are loaded)

This method is called once when an object with deferred validation (e.g. ModelEntities) become valid (usually after their resources have finished loading). Any deferred messages will be processed here.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.
get the class FourCC code

Get the class FourCC of the object.

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**Lighting::LightPrePassServer**
Lighting::LightPrePassServer Class Reference

#include <lightprepassserver.h>

Inheritance diagram for Lighting::LightPrePassServer:
Detailed Description

A LightServer which implements pre-light-pass rendering. Check here for details:
http://diaryofagraphicsprogrammer.blogspot.com/2008/03/light-pre-pass-renderer.html

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LightPrePassServer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~LightPrePassServer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>void Open ()</strong></td>
<td>Open the light server</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>Close the light server</td>
</tr>
<tr>
<td><strong>bool NeedsLightModelLinking ()</strong></td>
<td>Pre-light-pass doesn't require light/model linking</td>
</tr>
<tr>
<td><strong>void AttachVisibleLight (const Ptr&lt;InternalAbstractLightEntity&gt; &amp;lightEntity)</strong></td>
<td>Attach a visible light source</td>
</tr>
<tr>
<td><strong>void EndFrame ()</strong></td>
<td>End lighting frame</td>
</tr>
<tr>
<td><strong>void RenderLights ()</strong></td>
<td>Render light pass</td>
</tr>
<tr>
<td><strong>bool IsOpen ()</strong></td>
<td>Return true if light server is open</td>
</tr>
<tr>
<td><strong>void BeginFrame (const Ptr<a href="">InternalGraphics::InternalCameraEntity</a> &amp;cameraEntity)</strong></td>
<td>Begin lighting frame</td>
</tr>
<tr>
<td><strong>void BeginAttachVisibleLights ()</strong></td>
<td>Begin attaching visible light sources</td>
</tr>
<tr>
<td><strong>void EndAttachVisibleLights ()</strong></td>
<td>End attaching visible light sources</td>
</tr>
<tr>
<td><strong>void ApplyModelEntityLights (const Ptr<a href="">InternalGraphics::InternalModelEntity</a> &amp;modelEntity)</strong></td>
<td>Apply lighting parameters for a visible model entity</td>
</tr>
<tr>
<td><strong>int GetRefCount ()</strong></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName()</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC()</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><code>DumpRefCountingLeaks()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
**Protected Member Functions**

<table>
<thead>
<tr>
<th>void</th>
<th><strong>RenderGlobalLight</strong> ()</th>
<th>render the global light</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>RenderPointLights</strong> ()</td>
<td>render all point lights</td>
</tr>
<tr>
<td>void</td>
<td><strong>RenderSpotLights</strong> ()</td>
<td>render all spot lights</td>
</tr>
<tr>
<td>void</td>
<td><strong>AssignRenderBufferTextures</strong> ()</td>
<td>assign render buffers to shaders (one time init)</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```
bool Lighting::LightPrePassServer::NeedsLightModelLinking() const

pre-light-pass doesn't require light/model linking

Indicate whether the light server requires visibility links between lights and models. Deferred lighting solutions usually don't need this. FIXME: EXCEPT for shadow map rendering!!

Reimplemented from **Lighting::LightServerBase**.
```

```
void Lighting::LightPrePassServer::RenderGlobalLight() [protected]

render the global light

Render the global light as a fullscreen quad.
```

```
void Lighting::LightPrePassServer::RenderPointLights() [protected]

render all point lights

Render point lights into the LightBuffer.
```

```
void Lighting::LightPrePassServer::AssignRenderBufferTextures() [protected]

assign render buffers to shaders (one time init)

This does a one-time init of setting the NormalDepthBuffer and LightBuffer as shader variables.
```

```
void Lighting::LightServerBase::ApplyModelEntityLights(const Ptr<InternalGraphics::InternalModelEntity> moc &

apply lighting parameters for a visible model entity
```
This method is called during rendering to apply lighting parameters to the provided ModelEntity.

Reimplemented in \texttt{Lighting::SM30LightServer}.

\begin{verbatim}
int Core::RefCounted::GetRefCount() const [inline, inherited]
\end{verbatim}

get the current refcount

\begin{verbatim}
void Core::RefCounted::AddRef() [inline, inherited]
\end{verbatim}

increment refcount by one

\begin{verbatim}
void Core::RefCounted::Release() [inline, inherited]
\end{verbatim}

decrement refcount and destroy object if refcount is zero

\begin{verbatim}
const \texttt{Util::String} & Core::RefCounted::GetClassName() const [inline, inherited]
\end{verbatim}

get the class name

\begin{verbatim}
const \texttt{Util::FourCC} & Core::RefCounted::GetClassFourCC() const [inline, inherited]
\end{verbatim}

get the class FourCC code

\begin{verbatim}
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
\end{verbatim}
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**Lighting::LightServer**
#include <lightserver.h>

Inheritance diagram for Lighting::LightServer:

```
Core::RefCounted
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Lighting::LightServerBase</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Lighting::LightPrePassServer</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Lighting::LightServer</td>
</tr>
</tbody>
</table>
```

Lighting::LightServer Class Reference
Detailed Description

The light server collects all lights contributing to the scene and controls the realtime lighting process.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LightServer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~LightServer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Open ()</strong></td>
<td>open the light server</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>close the light server</td>
</tr>
<tr>
<td><strong>NeedsLightModelLinking ()</strong></td>
<td>const pre-light-pass doesn't require light/model linking</td>
</tr>
<tr>
<td><strong>AttachVisibleLight (const Ptr&lt;InternalAbstractLightEntity&gt; &amp;lightEntity)</strong></td>
<td>attach a visible light source</td>
</tr>
<tr>
<td><strong>EndFrame ()</strong></td>
<td>end lighting frame</td>
</tr>
<tr>
<td><strong>RenderLights ()</strong></td>
<td>render light pass</td>
</tr>
<tr>
<td><strong>IsOpen ()</strong></td>
<td>const return true if light server is open</td>
</tr>
<tr>
<td><strong>BeginFrame (const Ptr<a href="">InternalGraphics::InternalCameraEntity</a> &amp;cameraEntity)</strong></td>
<td>begin lighting frame</td>
</tr>
<tr>
<td><strong>BeginAttachVisibleLights ()</strong></td>
<td>begin attaching visible light sources</td>
</tr>
<tr>
<td><strong>EndAttachVisibleLights ()</strong></td>
<td>end attaching visible light sources</td>
</tr>
<tr>
<td><strong>ApplyModelEntityLights (const Ptr<a href="">InternalGraphics::InternalModelEntity</a> &amp;modelEntity)</strong></td>
<td>apply lighting parameters for a visible model entity</td>
</tr>
<tr>
<td><strong>GetRefCount ()</strong></td>
<td>const get the current refcount</td>
</tr>
</tbody>
</table>
void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className)
const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC)
const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void RenderGlobalLight()</td>
<td>render the global light</td>
</tr>
<tr>
<td>void RenderPointLights()</td>
<td>render all point lights</td>
</tr>
<tr>
<td>void RenderSpotLights()</td>
<td>render all spot lights</td>
</tr>
<tr>
<td>void AssignRenderBufferTextures()</td>
<td>assign render buffers to shaders (one time init)</td>
</tr>
</tbody>
</table>
bool Lighting::LightPrePassServer::NeedsLightModelLinking() const [inherited]

pre-light-pass doesn't require light/model linking

Indicate whether the light server requires visibility links between lights and models. Deferred lighting solutions usually don't need this.
FIXME: EXCEPT for shadow map rendering!!

Reimplemented from Lighting::LightServerBase.

void Lighting::LightPrePassServer::RenderGlobalLight() [protected, inherited]

render the global light

Render the global light as a fullscreen quad.

void Lighting::LightPrePassServer::RenderPointLights() [protected, inherited]

render all point lights

Render point lights into the LightBuffer.

void Lighting::LightPrePassServer::AssignRenderBufferTextures() [protected, inherited]

assign render buffers to shaders (one time init)

This does a one-time init of setting the NormalDepthBuffer and LightBuffer as shader variables.

void Lighting::LightServerBase::ApplyModelEntityLights(const Ptr<InternalGraphics::InternalModelEntity> & moc)

apply lighting parameters for a visible model entity
This method is called during rendering to apply lighting parameters to the provided ModelEntity.

Reimplemented in `Lighting::SM30LightServer`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Lighting::LightServerBase
#include <lightserverbase.h>

Inheritance diagram for Lighting::LightServerBase:
**Detailed Description**

**Base** class for the light server. The light server collects all lights contributing to the scene. Subclasses of **LightServerBase** implement specific lighting techniques.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LightServerBase ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~LightServerBase ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void Open ()</td>
<td>open the light server</td>
</tr>
<tr>
<td>void Close ()</td>
<td>close the light server</td>
</tr>
<tr>
<td>bool IsOpen () const</td>
<td>return true if light server is open</td>
</tr>
<tr>
<td>bool NeedsLightModelLinking () const</td>
<td>indicate whether the light server requires light/model linking</td>
</tr>
</tbody>
</table>
| void BeginFrame (const Ptr<
  InternalGraphics::InternalCameraEntity > &cameraEntity) | begin lighting frame                                                      |
| void BeginAttachVisibleLights ()             | begin attaching visible light sources                                       |
| void AttachVisibleLight (const Ptr<
  InternalAbstractLightEntity > &lightEntity) | attach a visible light source                                              |
| void EndAttachVisibleLights ()               | end attaching visible light sources                                        |
| void ApplyModelEntityLights (const Ptr<
  InternalGraphics::InternalModelEntity > &modelEntity) | apply lighting parameters for a visible model entity                      |
<p>| void RenderLights ()                         | render light pass                                                          |
| void EndFrame ()                             | end lighting frame                                                         |
| int GetRefCount () const                     | get the current refcount                                                   |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName() const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC() const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

bool Lighting::LightServerBase::NeedsLightModelLinking() const

indicate whether the light server requires light/model linking

Indicate whether the light server requires visibility links between lights and models. Deferred lighting solutions usually don't need this.

FIXME: EXCEPT for shadow map rendering!!

Reimplemented in **Lighting::LightPrePassServer**.

```cpp
void Lighting::LightServerBase::ApplyModelEntityLights(const Ptr<InternalGraphics::InternalModelEntity> &modelEntity)
```

apply lighting parameters for a visible model entity

This method is called during rendering to apply lighting parameters to the provided ModelEntity.

Reimplemented in **Lighting::SM30LightServer**.

```cpp
void Lighting::LightServerBase::RenderLights()
```

render light pass

This method is called when a FrameBatch of type **Lighting** is rendered.

Reimplemented in **Lighting::LightPrePassServer**.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.
increment refcount by one

Increment the refcount of the object.

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

get the class name

Get the class name of the object.

generate refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Lighting::LightType
Lighting::LightType Class Reference

#include <lighttype.h>
Detailed Description

Identifies different light types.

(C) 2007 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static Code</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FromString</strong></td>
<td>(const <strong>Util::String</strong> &amp;str)</td>
<td>convert from string</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>(Code code)</td>
<td>convert to string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by `doxygen` at Fri Mar 26 15:21:47 2010
Lighting::PSSMUtil
Lighting::PSSMUtil Class Reference

#include <pssmutil.h>
Detailed Description

Helper class which compute LightProj matrices for Parallel-Split-Shadowmap rendering.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PSSMUtil</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td><strong>SetCameraEntity</strong> (const Ptr<a href="">InternalGraphics::InternalCameraEntity</a> &amp;camera)</td>
<td>set camera entity which defines the view transform</td>
</tr>
<tr>
<td><strong>GetCameraEntity</strong> () const</td>
<td>get camera entity</td>
</tr>
<tr>
<td><strong>SetLightDir</strong> (const Math::vector &amp;)</td>
<td>set light direction</td>
</tr>
<tr>
<td><strong>GetLightDir</strong> () const</td>
<td>get light direction</td>
</tr>
<tr>
<td><strong>Compute</strong> ()</td>
<td>compute PSSM split volumes</td>
</tr>
<tr>
<td><strong>GetSplitLightTransform</strong> (IndexT splitIndex) const</td>
<td>get view matrix for a view frustum split (valid after Compute)</td>
</tr>
<tr>
<td><strong>GetSplitProjTransform</strong> (IndexT splitIndex) const</td>
<td>get projection transform for a view frustum split (valid after Compute)</td>
</tr>
<tr>
<td><strong>GetSplitLightProjTransform</strong> (IndexT splitIndex) const</td>
<td>get light projection transform for given frustum split (valid after Compute)</td>
</tr>
<tr>
<td><strong>GetSplitDistances</strong> () const</td>
<td>get raw pointer to split distances</td>
</tr>
<tr>
<td><strong>GetSplitLightProjTransforms</strong></td>
<td>get raw pointer to LightProjTransforms</td>
</tr>
</tbody>
</table>
Static Public Attributes

static const SizeT NumSplits = 4

number of view volume splits
Lighting::ShadowServer
Lighting::ShadowServer Class Reference

#include <shadowserver.h>

Inheritance diagram for Lighting::ShadowServer:

Lighting::SM30ShadowServer

Lighting::ShadowServer
Detailed Description

The **ShadowServer** setups and controls the global aspects of the dynamic shadow system.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShadowServer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~ShadowServer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Open ()</strong></td>
<td>open the shadow server</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close the shadow server</td>
</tr>
<tr>
<td><strong>void UpdateShadowBuffers ()</strong></td>
<td>update shadow buffer</td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::Texture &gt;&amp;</strong></td>
<td>get pointer to shadow buffer for local lights</td>
</tr>
<tr>
<td><strong>GetLocalLightShadowBufferTexture () const</strong></td>
<td></td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::Texture &gt;&amp;</strong></td>
<td>get pointer to PSSM shadow buffer for global lights</td>
</tr>
<tr>
<td><strong>GetGlobalLightShadowBufferTexture () const</strong></td>
<td></td>
</tr>
<tr>
<td><strong>const float * ** GetPSSMSplitDistances () const</strong></td>
<td>get array of PSSM split distances</td>
</tr>
<tr>
<td><strong>const Math::matrix44 * ** GetPSSMSplitLightProjTransforms () const</strong></td>
<td>get array of PSSM LightProjTransforms</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Lighting::SM30ShadowServer::UpdateShadowBuffers ( ) [inherited]

update shadow buffer

This method updates the internal shadow buffer render targets.

const float *
Lighting::SM30ShadowServer::GetPSSMSplitDistances ( ) const [inherited]

get array of PSSM split distances

Get raw pointer to array of PSSM split distances.

const Math::matrix44 *
Lighting::SM30ShadowServer::GetPSSMSplitLightProjTransforms ( ) const [inherited]

get array of PSSM LightProjTransforms

Get raw pointer to array of PSSM split LightProjTransform matrices.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:47 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Lighting::SM30LightServer
Lighting::SM30LightServer Class Reference

#include <sm30lightserver.h>

Inheritance diagram for Lighting::SM30LightServer:
Detailed Description

The standard light server for platforms which are capable of ShaderModel 3.0 or better.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SM30LightServer()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~SM30LightServer()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>Open the light server</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>Close the light server</td>
</tr>
<tr>
<td><code>ApplyModelEntityLights</code></td>
<td>Apply lighting parameters for a visible model entity</td>
</tr>
<tr>
<td><code>IsOpen()</code></td>
<td>Return true if light server is open</td>
</tr>
<tr>
<td><code>NeedsLightModelLinking()</code></td>
<td>Indicate whether the light server requires light/model linking</td>
</tr>
<tr>
<td><code>BeginFrame</code></td>
<td>Begin lighting frame</td>
</tr>
<tr>
<td><code>BeginAttachVisibleLights()</code></td>
<td>Begin attaching visible light sources</td>
</tr>
<tr>
<td><code>AttachVisibleLight</code></td>
<td>Attach a visible light source</td>
</tr>
<tr>
<td><code>EndAttachVisibleLights()</code></td>
<td>End attaching visible light sources</td>
</tr>
<tr>
<td><code>RenderLights()</code></td>
<td>Render light pass</td>
</tr>
<tr>
<td><code>EndFrame()</code></td>
<td>End lighting frame</td>
</tr>
<tr>
<td><code>GetRefCount()</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>Get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>Get the class FourCC code</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>static void <strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void Lighting::SM30LightServer::ApplyModelEntityLights(const Ptr<InternalGraphics::InternalModelEntity> &modelEntity)
```

apply lighting parameters for a visible model entity

**Todo:**
set light properties only once per-frame and only set a bool array with active per-model-entity-lights here!

Reimplemented from `Lighting::LightServerBase`.

```cpp
bool Lighting::LightServerBase::NeedsLightModelLinking()
```

indicate whether the light server requires light/model linking

Indicate whether the light server requires visibility links between lights and models. Deferred lighting solutions usually don't need this. FIXME: EXCEPT for shadow map rendering!!

Reimplemented in `Lighting::LightPrePassServer`.

```cpp
void Lighting::LightServerBase::RenderLights()
```

render light pass

This method is called when a FrameBatch of type `Lighting` is rendered.

Reimplemented in `Lighting::LightPrePassServer`.

```cpp
int Core::RefCounted::GetRefCount()
```

get the current refcount
Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Lighting::SM30ShadowServer
Lighting::SM30ShadowServer Class Reference

#include <sm30shadowserver.h>

Inheritance diagram for Lighting::SM30ShadowServer:
Detailed Description

Shadow server for platforms which support at least shader model 3.0. Implements simple shadow mapping for local lights and parallel-split-shadow-mapping for the global light source

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SM30ShadowServer()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~SM30ShadowServer()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>void Open()</strong></td>
<td>Open the shadow server</td>
</tr>
<tr>
<td><strong>void Close()</strong></td>
<td>Close the shadow server</td>
</tr>
<tr>
<td><strong>void UpdateShadowBuffers()</strong></td>
<td>Update shadow buffer</td>
</tr>
<tr>
<td><strong>const Ptr<a href="">CoreGraphics::Texture</a> &amp;</strong></td>
<td>GetLocalLightShadowBufferTexture() const</td>
</tr>
<tr>
<td><strong>get pointer to shadow buffer for local lights</strong></td>
<td></td>
</tr>
<tr>
<td><strong>const Ptr<a href="">CoreGraphics::Texture</a> &amp;</strong></td>
<td>GetGlobalLightShadowBufferTexture() const</td>
</tr>
<tr>
<td><strong>get pointer to PSSM shadow buffer for global lights</strong></td>
<td></td>
</tr>
<tr>
<td><strong>const float * GetPSSMSplitDistances()</strong></td>
<td>GetPSSMSplitDistances() const</td>
</tr>
<tr>
<td><strong>get array of PSSM split distances</strong></td>
<td></td>
</tr>
<tr>
<td><strong>const Math::matrix44 * GetPSSMSplitLightProjTransforms()</strong></td>
<td>GetPSSMSplitLightProjTransforms() const</td>
</tr>
<tr>
<td><strong>get array of PSSM LightProjTransforms</strong></td>
<td></td>
</tr>
</tbody>
</table>
void Lighting::SM30ShadowServer::UpdateShadowBuffers()

update shadow buffer

This method updates the internal shadow buffer render taregets.

const float * Lighting::SM30ShadowServer::GetPSSMSplitDistances()

get array of PSSM split distances

Get raw pointer to array of PSSM split distances.

const Math::matrix44 * Lighting::SM30ShadowServer::GetPSSMSplitLightProjTransforms()

get array of PSSM LightProjTransforms

Get raw pointer to array of PSSM split LightProjTransform matrices.
Managers::LogManager
Managers::LogManager Class Reference

#include <logmanager.h>
Detailed Description

The log manager manages log entries in a ring buffer.

You can add log entries (e.g. in form of a info log object) and you can get all log entries currently in the ring buffer or new log entries which have been added since the last getting of log entries.

A log manager uses a log user interface which will be updated when new log entries are added.

The initial capacity of the managed ring buffer will be 4000 log entries.

(C) 2006 Radon Labs GmbH
Managers::\texttt{QuestManager}
Managers::QuestManager Class Reference

#include <questmanager.h>
Detailed Description

Provides access to the quest system, can list all quest in their various states, can unlock quests, etc...

A complete overview of the quest system can be found here: http://gambar/wiki/index.php/DSA_Story_Subsystem

(C) 2005 Radon Labs GmbH
Managers::<strong>ScriptTable</strong>
Managers::ScriptTable Class Reference

#include <scripttable.h>
Detailed Description

(C) 2007 Radon Labs GmbH
Managers::ScriptTemplateManager
Managers::ScriptTemplateManager
Class Reference

#include <scriptTEMPLATEmANAGER.h>
Detailed Description

Wraps entity categories and provides access to category template and instance tables.

(C) 2007 Radon Labs GmbH
Math::bbox
Math::bbox Class Reference

#include <bbox.h>
Detailed Description

Nebula's bounding box class.

Todo:
   : UNTESTED!

(C) 2004 RadonLabs GmbH
**Public Types**

<table>
<thead>
<tr>
<th>enum</th>
<th>clip codes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>clip codes</strong></td>
<td></td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bbox()</code></td>
<td>Constructor 1</td>
</tr>
<tr>
<td><code>bbox(const point &amp;center, const vector &amp;extents)</code></td>
<td>Constructor 3</td>
</tr>
<tr>
<td><code>bbox(const matrix44 &amp;m)</code></td>
<td>Construct bounding box from <code>matrix44</code></td>
</tr>
<tr>
<td><code>point center()</code></td>
<td>Get center of box</td>
</tr>
<tr>
<td><code>vector extents()</code></td>
<td>Get extents of box</td>
</tr>
<tr>
<td><code>vector size()</code></td>
<td>Get size of box</td>
</tr>
<tr>
<td><code>scalar diagonal_size()</code></td>
<td>Get diagonal size of box</td>
</tr>
<tr>
<td><code>void set(const matrix44 &amp;m)</code></td>
<td>Set from <code>matrix44</code></td>
</tr>
<tr>
<td><code>void set(const point &amp;center, const vector &amp;extents)</code></td>
<td>Set from center point and extents</td>
</tr>
<tr>
<td><code>void begin_extend()</code></td>
<td>Begin extending the box</td>
</tr>
<tr>
<td><code>void extend(const point &amp;p)</code></td>
<td>Extend the box</td>
</tr>
<tr>
<td><code>void extend(const bbox &amp;box)</code></td>
<td>Extend the box</td>
</tr>
<tr>
<td><code>void end_extend()</code></td>
<td>This resets the bounding box size to zero if no <code>extend()</code> method was called after <code>begin_extend()</code></td>
</tr>
<tr>
<td><code>void transform(const matrix44 &amp;m)</code></td>
<td>Transform bounding box</td>
</tr>
<tr>
<td><code>bool intersects(const bbox &amp;box)</code> const</td>
<td>Check for intersection with axis aligned bounding box</td>
</tr>
<tr>
<td><code>bool contains(const bbox &amp;box)</code> const</td>
<td>Check if this box completely contains the parameter box</td>
</tr>
<tr>
<td>bool contains (const point &amp;p) const</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td></td>
</tr>
<tr>
<td>return true if this box contains the position</td>
<td></td>
</tr>
</tbody>
</table>

ClipStatus::Type clipstatus (const bbox &other) const
check for intersection with other bounding box

ClipStatus::Type clipstatus (const matrix44 &viewProjection) const
check for intersection with projection volume

matrix44 to_matrix44 () const
create a matrix which transforms a unit cube to this bounding box

point corner_point (int index) const
return one of the 8 corner points

void get_clipplanes (const matrix44 &viewProjection,
Util::Array< plane > &outPlanes) const
return side planes in clip space

template<typename T>
T as () const
convert to any type
Member Function Documentation

```cpp
void Math::bbox::set(const matrix44 m) [inline]
```

set from `matrix44`

Construct a bounding box around a 4x4 matrix. The translational part defines the center point, and the x,y,z vectors of the matrix define the extents.

```cpp
void Math::bbox::endExtend() [inline]
```

this resets the bounding box size to zero if no `extend()` method was called after `begin_extend()`

This just checks whether the `extend()` method has actually been called after `begin_extend()` and just sets `vmin` and `vmax` to the null vector if it hasn't.

```cpp
void Math::bbox::transform(const matrix44 m) [inline]
```

transform bounding box

Transforms this axis aligned bounding by the 4x4 matrix. This bounding box must be axis aligned with the matrix, the resulting bounding will be axis aligned in the matrix' "destination" space.

E.g. if you have a bounding box in model space 'modelBox', and a 'modelView' matrix, the operation

```
modelBox.transform(modelView)
```

would transform the bounding box into view space.
Math::bbox::intersects (bbox box) const [inline]
&

check for intersection with axis aligned bounding box

Check for intersection of 2 axis aligned bounding boxes. The bounding boxes must live in the same coordinate space.

bool Math::bbox::contains (bbox box) const [inline]
&

check if this box completely contains the parameter box

Check if the parameter bounding box is completely contained in this bounding box.

bool Math::bbox::contains (point p) const [inline]
&

return true if this box contains the position

Check if position is inside bounding box.

ClipStatus::Type Math::bbox::clipstatus (bbox other) const
&

check for intersection with other bounding box

Return box/box clip status.

ClipStatus::Type Math::bbox::clipstatus (matrix44 viewProjection) const
&

check for intersection with projection volume

Check for intersection with a view volume defined by a view-projection matrix.

matrix44 () const [inline]
Math::bbox::to_matrix44

create a matrix which transforms a unit cube to this bounding box

Create a transform matrix which would transform a unit cube to this bounding box.

point
Math::bbox::corner_point(int index) const

return one of the 8 corner points

Returns one of the 8 corners of the bounding box.

void
Math::bbox::get_clipplanes(const matrix44 & viewProj,
    Util::Array<plane> & outPlanes
)

return side planes in clip space

Get the bounding box's side planes in clip space.
Math::ClipStatus
Math::ClipStatus Class Reference

#include <clipstatus.h>
Detailed Description

The result of a clip check (Inside, Outside or Clipped).

(C) 2007 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Math::Extrapolator
Math::Extrapolator< TYPE > Class Template Reference

#include <extrapolator.h>
Detailed Description

template<class TYPE>
class Math::Extrapolator< TYPE >

Extrapolator maintains state about updates for remote entities, and will generate smooth guesses about where those entities will be based on previously received data.

This implementation is based on http://www.mindcontrol.org/~hplus/epic/ Its adapted for use with point and vector class.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Extrapolator</strong> ()</td>
<td>Constructor</td>
<td></td>
</tr>
<tr>
<td>virtual <strong>~Extrapolator</strong> ()</td>
<td>Destructor</td>
<td></td>
</tr>
<tr>
<td><strong>bool AddSample</strong> (Timing::Time packetTime, Timing::Time curTime, const TYPE &amp;pos)</td>
<td>Add sample without velocity, velocity is compute from positions</td>
<td></td>
</tr>
<tr>
<td><strong>bool AddSample</strong> (Timing::Time packetTime, Timing::Time curTime, const TYPE &amp;pos, const TYPE &amp;vel)</td>
<td>Add sample with given velocity</td>
<td></td>
</tr>
<tr>
<td><strong>void Reset</strong> (Timing::Time packetTime, Timing::Time curTime, const TYPE &amp;pos)</td>
<td>Re-set the Extrapolator's idea of time, velocity and position.</td>
<td></td>
</tr>
<tr>
<td><strong>void Reset</strong> (Timing::Time packetTime, Timing::Time curTime, const TYPE &amp;pos, const TYPE &amp;vel)</td>
<td>Re-set the Extrapolator's idea of time, velocity and position.</td>
<td></td>
</tr>
<tr>
<td><strong>bool ReadValue</strong> (Timing::Time forTime, TYPE &amp;oPos) const</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>bool ReadValue</strong> (Timing::Time forTime, TYPE &amp;oPos, TYPE &amp;oVel) const</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Timing::Time EstimateLatency</strong> () const</td>
<td>Returns: the current estimation of latency between the sender and this interpolator.</td>
<td></td>
</tr>
<tr>
<td><strong>Timing::Time EstimateUpdateTime</strong> () const</td>
<td>Returns: the current estimation of frequency of updates that the sender will send.</td>
<td></td>
</tr>
</tbody>
</table>


Member Function Documentation

template<class TYPE>
bool Math::Extrapolator< ( Timing::Time forTime,
TYPE >::ReadValue
    TYPE & oPos
) const

Return an estimate of the interpolated position at a given global time
(which typically will be greater than the curTime passed into
AddSample()).

template<class TYPE>
bool Math::Extrapolator< ( Timing::Time forTime,
TYPE >::ReadValue
    TYPE & oPos,
    TYPE & oVel
) const

Return an estimate of the interpolated position at a given global time
(which typically will be greater than the curTime passed into
AddSample()).
Math::float2
Math::float2 Class Reference

#include <float2.h>
Detailed Description

A 2-component float vector class.

(C) 2007 RadonLabs GmbH
Public Member Functions

- float2 ()
  default constructor, NOTE: does NOT setup components!

- float2 (scalar x, scalar y)
  construct from values

- float2 (const float2 &rhs)
  copy constructor

- void operator= (const float2 &rhs)
  assignment operator

- float2 operator- () const
  flip sign

- void operator+= (const float2 &rhs)
  inplace add

- void operator-= (const float2 &rhs)
  inplace sub

- void operator*=(scalar s)
  inplace scalar multiply

- float2 operator+ (const float2 &rhs) const
  add 2 vectors

- float2 operator- (const float2 &rhs) const
  subtract 2 vectors

- float2 operator* (scalar s) const
  multiply with scalar

- bool operator== (const float2 &rhs) const
  equality operator

- bool operator!= (const float2 &rhs) const
  inequality operator

- void set (scalar x, scalar y)
  set content

- scalar & x ()
  read/write access to x component

- scalar & y ()
  read/write access to y component

- scalar x () const
**read-only access to x component**

**vector**

**scalar**

**y () const**

*read-only access to y component*

**scalar**

**length () const**

*return length of vector*

**scalar**

**lengthsq () const**

*return squared length of vector*

**float2**

**abs () const**

*return component-wise absolute*

**bool**

**any () const**

*return true if any components are non-zero*

**bool**

**all () const**

*return true if all components are non-zero*

**template<typename T>**

**T as () const**

*convert to anything*
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>maximize</code></td>
<td><code>static float2 maximize (const float2 &amp;v0, const float2 &amp;v1)</code> return vector made up of largest components of 2 vectors</td>
</tr>
<tr>
<td><code>minimize</code></td>
<td><code>static float2 minimize (const float2 &amp;v0, const float2 &amp;v1)</code> return vector made up of smallest components of 2 vectors</td>
</tr>
<tr>
<td><code>normalize</code></td>
<td><code>static float2 normalize (const float2 &amp;v)</code> return normalized version of vector</td>
</tr>
<tr>
<td><code>lt</code></td>
<td><code>static float2 lt (const float2 &amp;v0, const float2 &amp;v1)</code> set less-than components to non-zero</td>
</tr>
<tr>
<td><code>le</code></td>
<td><code>static float2 le (const float2 &amp;v0, const float2 &amp;v1)</code> set less-or-equal components to non-zero</td>
</tr>
<tr>
<td><code>gt</code></td>
<td><code>static float2 gt (const float2 &amp;v0, const float2 &amp;v1)</code> set greater-than components to non-zero</td>
</tr>
<tr>
<td><code>ge</code></td>
<td><code>static float2 ge (const float2 &amp;v0, const float2 &amp;v1)</code> set greater-or-equal components to non-zero</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:47 2010
Math::float4
#include <xna_float4.h>

Inheritance diagram for Math::float4:
Detailed Description

A 4-component float vector class. This is the basis class for points and vectors.

(C) 2007 RadonLabs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>float4 ()</strong></td>
<td>default constructor, NOTE: does NOT setup components!</td>
</tr>
<tr>
<td><strong>float4 (scalar x, scalar y, scalar z, scalar w)</strong></td>
<td>construct from values</td>
</tr>
<tr>
<td><strong>float4 (XMVECTOR rhs)</strong></td>
<td>construct from XMVECTOR</td>
</tr>
<tr>
<td><strong>void operator= (const float4 &amp;rhs)</strong></td>
<td>assignment operator</td>
</tr>
<tr>
<td><strong>void operator= (XMVECTOR rhs)</strong></td>
<td>assign an XMVECTOR</td>
</tr>
<tr>
<td><strong>float4 operator- () const</strong></td>
<td>flip sign</td>
</tr>
<tr>
<td><strong>void operator+= (const float4 &amp;rhs)</strong></td>
<td>inplace add</td>
</tr>
<tr>
<td><strong>void operator-= (const float4 &amp;rhs)</strong></td>
<td>inplace sub</td>
</tr>
<tr>
<td>*<em>void operator <em>= (scalar s)</em></em></td>
<td>inplace scalar multiply</td>
</tr>
<tr>
<td>*<em>void operator <em>= (const float4 &amp;rhs)</em></em></td>
<td>multiply by a vector component-wise</td>
</tr>
<tr>
<td><strong>float4 operator+ (const float4 &amp;rhs) const</strong></td>
<td>add 2 vectors</td>
</tr>
<tr>
<td><strong>float4 operator- (const float4 &amp;rhs) const</strong></td>
<td>subtract 2 vectors</td>
</tr>
<tr>
<td><strong>float4 operator * (scalar s) const</strong></td>
<td>multiply with scalar</td>
</tr>
<tr>
<td><strong>bool operator== (const float4 &amp;rhs) const</strong></td>
<td>equality operator</td>
</tr>
<tr>
<td><strong>bool operator!= (const float4 &amp;rhs) const</strong></td>
<td>inequality operator</td>
</tr>
<tr>
<td>*<em>void load (const scalar <em>ptr)</em></em></td>
<td>load content from 16-byte-aligned memory</td>
</tr>
<tr>
<td>*<em>void loadu (const scalar <em>ptr)</em></em></td>
<td></td>
</tr>
</tbody>
</table>
load content from unaligned memory

void **store** (scalar *ptr) const
write content to 16-byte-aligned memory through the write cache

void **storeu** (scalar *ptr) const
write content to unaligned memory through the write cache

void **stream** (scalar *ptr) const
stream content to 16-byte-aligned memory circumventing the write-cache

void **load_float3** (const void *ptr, float w)
load 3 floats into x,y,z from unaligned memory

void **load_ubyte4n_signed** (const void *ptr, float w)
load from UByte4N packed vector, move range to -1..+1

void **set** (scalar x, scalar y, scalar z, scalar w)
set content

void **set_x** (scalar x)
set the x component

void **set_y** (scalar y)
set the y component

void **set_z** (scalar z)
set the z component

void **set_w** (scalar w)
set the w component

scalar & **x** ()
read/write access to x component

scalar & **y** ()
read/write access to y component

scalar & **z** ()
read/write access to z component

scalar & **w** ()
read/write access to w component

scalar **x** () const
read-only access to x component

scalar **y** () const
read-only access to y component

scalar **z** () const
read-only access to z component

scalar **w** () const
read-only access to w component
<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scalar</td>
<td>length()</td>
<td>const return length of vector</td>
</tr>
<tr>
<td>scalar</td>
<td>lengthsq()</td>
<td>const return squared length of vector</td>
</tr>
<tr>
<td>float4</td>
<td>abs()</td>
<td>const return component-wise absolute</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>reciprocal</code> (const <code>float4</code> &amp;v)</td>
<td>return $1.0 / vec</td>
</tr>
<tr>
<td><code>multiply</code> (const <code>float4</code> &amp;v0, const <code>float4</code> &amp;v1)</td>
<td>component-wise multiplication</td>
</tr>
<tr>
<td><code>cross3</code> (const <code>float4</code> &amp;v0, const <code>float4</code> &amp;v1)</td>
<td>return 3-dimensional cross product</td>
</tr>
<tr>
<td><code>dot3</code> (const <code>float4</code> &amp;v0, const <code>float4</code> &amp;v1)</td>
<td>return 3d dot product of vectors</td>
</tr>
<tr>
<td><code>barycentric</code> (const <code>float4</code> &amp;v0, const <code>float4</code> &amp;v1, const <code>float4</code> &amp;v2, scalar f, scalar g)</td>
<td>return point in barycentric coordinates</td>
</tr>
<tr>
<td><code>catmullrom</code> (const <code>float4</code> &amp;v0, const <code>float4</code> &amp;v1, const <code>float4</code> &amp;v2, const <code>float4</code> &amp;v3, scalar s)</td>
<td>perform Catmull-Rom interpolation</td>
</tr>
<tr>
<td><code>hermite</code> (const <code>float4</code> &amp;v1, const <code>float4</code> &amp;t1, const <code>float4</code> &amp;v2, const <code>float4</code> &amp;t2, scalar s)</td>
<td>perform Hermite spline interpolation</td>
</tr>
<tr>
<td><code>lerp</code> (const <code>float4</code> &amp;v0, const <code>float4</code> &amp;v1, scalar s)</td>
<td>perform linear interpolation between 2 4d vectors</td>
</tr>
<tr>
<td><code>maximize</code> (const <code>float4</code> &amp;v0, const <code>float4</code> &amp;v1)</td>
<td>return 4d vector made up of largest components of 2 vectors</td>
</tr>
<tr>
<td><code>minimize</code> (const <code>float4</code> &amp;v0, const <code>float4</code> &amp;v1)</td>
<td>return 4d vector made up of smallest components of 2 vectors</td>
</tr>
<tr>
<td><code>normalize</code> (const <code>float4</code> &amp;v)</td>
<td>return normalized version of 4d vector</td>
</tr>
<tr>
<td><code>transform</code> (const <code>float4</code> &amp;v)</td>
<td>transform 4d vector by <code>matrix44</code></td>
</tr>
<tr>
<td><code>reflect</code> (const <code>float4</code> &amp;normal, const <code>float4</code> &amp;incident)</td>
<td>reflect a vector v</td>
</tr>
<tr>
<td><code>clamp</code> (const <code>float4</code> &amp;vMin, const <code>float4</code> &amp;vMax)</td>
<td>clamp to min/max vector</td>
</tr>
<tr>
<td><code>angle</code> (const <code>float4</code> &amp;v0, const <code>float4</code> &amp;v1)</td>
<td></td>
</tr>
</tbody>
</table>
angle between two vectors

static bool **less3**_any (const float &v0, const float &v1)
return true if any XYZ component is less-than

static bool **less3**_all (const float &v0, const float &v1)
return true if all XYZ components are less-than

static bool **lessequal3**_any (const float &v0, const float &v1)
return true if any XYZ component is less-or-equal

static bool **lessequal3**_all (const float &v0, const float &v1)
return true if all XYZ components are less-or-equal

static bool **greater3**_any (const float &v0, const float &v1)
return true if any XYZ component is greater

static bool **greater3**_all (const float &v0, const float &v1)
return true if all XYZ components are greater

static bool **greaterequal3**_any (const float &v0, const float &v1)
return true if any XYZ component is greater-or-equal

static bool **greaterequal3**_all (const float &v0, const float &v1)
return true if all XYZ components are greater-or-equal

static bool **equal3**_any (const float &v0, const float &v1)
return true if any XYZ component is equal

static bool **equal3**_all (const float &v0, const float &v1)
return true if all XYZ components are equal

static bool **nearequal3** (const float &v0, const float &v1, const float &epsilon)
perform near equal comparison with given epsilon (3 components)

static bool **less4**_any (const float &v0, const float &v1)
return true if any XYZW component is less-than

static bool **less4**_all (const float &v0, const float &v1)
return true if all XYZW components are less-than

static bool **lessequal4**_any (const float &v0, const float &v1)
return true if any XYZW component is less-or-equal

static bool **lessequal4**_all (const float &v0, const float &v1)
return true if all XYZW components are less-or-equal

static bool **greater4**_any (const float &v0, const float &v1)
return true if any XYZW component is greater

static bool **greater4**_all (const float &v0, const float &v1)
return true if all XYZW components are greater
<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>greaterequal4_any</code></td>
<td><code>const float4 &amp;v0, const float4 &amp;v1</code></td>
<td>Return true if any X, Y, Z, W component is greater-or-equal</td>
</tr>
<tr>
<td><code>greaterequal4_all</code></td>
<td><code>const float4 &amp;v0, const float4 &amp;v1</code></td>
<td>Return true if all X, Y, Z, W components are greater-or-equal</td>
</tr>
<tr>
<td><code>equal4_any</code></td>
<td><code>const float4 &amp;v0, const float4 &amp;v1</code></td>
<td>Return true if any X, Y, Z, W component is equal</td>
</tr>
<tr>
<td><code>equal4_all</code></td>
<td><code>const float4 &amp;v0, const float4 &amp;v1</code></td>
<td>Return true if all X, Y, Z, W components are equal</td>
</tr>
<tr>
<td><code>nearequal4</code></td>
<td><code>const float4 &amp;v0, const float4 &amp;v1, const float4 &amp;epsilon</code></td>
<td>Perform near equal comparison with given epsilon (4 components)</td>
</tr>
<tr>
<td><code>unpack_x</code></td>
<td><code>XMVECTOR v</code></td>
<td>Unpack the first element from a XMVECTOR</td>
</tr>
<tr>
<td><code>unpack_y</code></td>
<td><code>XMVECTOR v</code></td>
<td>Unpack the second element from a XMVECTOR</td>
</tr>
<tr>
<td><code>unpack_z</code></td>
<td><code>XMVECTOR v</code></td>
<td>Unpack the third element from a XMVECTOR</td>
</tr>
<tr>
<td><code>unpack_w</code></td>
<td><code>XMVECTOR v</code></td>
<td>Unpack the fourth element from a XMVECTOR</td>
</tr>
<tr>
<td><code>splat</code></td>
<td><code>scalar s</code></td>
<td>Splat scalar into each component of a vector</td>
</tr>
<tr>
<td><code>splat</code></td>
<td><code>const float4 &amp;v, uint element</code></td>
<td>Return a vector with all elements set to element n of v, 0 &lt;= element &lt;= 3</td>
</tr>
<tr>
<td><code>splat_x</code></td>
<td><code>const float4 &amp;v</code></td>
<td>Return a vector with all elements set to v.x</td>
</tr>
<tr>
<td><code>splat_y</code></td>
<td><code>const float4 &amp;v</code></td>
<td>Return a vector with all elements set to v.y</td>
</tr>
<tr>
<td><code>splat_z</code></td>
<td><code>const float4 &amp;v</code></td>
<td>Return a vector with all elements set to v.z</td>
</tr>
<tr>
<td><code>splat_w</code></td>
<td><code>const float4 &amp;v</code></td>
<td>Return a vector with all elements set to v.w</td>
</tr>
<tr>
<td><code>permute_control</code></td>
<td><code>unsigned int i0, unsigned int i1, unsigned int i2, unsigned int i3</code></td>
<td>Return control vector for permute (see XMVectorPermuteControl for details)</td>
</tr>
<tr>
<td><code>permute</code></td>
<td><code>const float4 &amp;v0, const float4 &amp;v1, const float4 &amp;control</code></td>
<td>Merge components of 2 vectors into 1 (see XMVectorPermute for details)</td>
</tr>
</tbody>
</table>
Constructor & Destructor Documentation

__forceinline
Math::float4::float4 ( XMVECTOR rhs )

construct from XMVECTOR

!!!! copy constructor forbidden, otherwise passing float4's to a function
!!!! via Registers doesn't work
Member Function Documentation

__forceinline void Math::float4::load ( scalar *ptr )

load content from 16-byte-aligned memory

Load 4 floats from 16-byte-aligned memory.

__forceinline void Math::float4::loadu ( scalar *ptr )

load content from unaligned memory

Load 4 floats from unaligned memory.

__forceinline void Math::float4::store ( *ptr ) const

write content to 16-byte-aligned memory through the write cache

Store to 16-byte-aligned float pointer.

__forceinline void Math::float4::storeu ( *ptr ) const

write content to unaligned memory through the write cache

Store to non-aligned float pointer.
Math::frustum
#include <frustum.h>
Detailed Description

Defines a clipping frustum made of 6 planes.

(C) 2010 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>PlaneIndex</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>plane indices</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>frustum ()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>frustum (const matrix44 &amp;invViewProj)</code></td>
<td>construct from view and projection matrix</td>
</tr>
<tr>
<td><code>void set (const matrix44 &amp;invViewProj)</code></td>
<td>setup from view and proj matrix</td>
</tr>
<tr>
<td><code>void set (const bbox &amp;box, const matrix44 &amp;boxTransform)</code></td>
<td>setup from transformed bounding box</td>
</tr>
<tr>
<td><code>bool inside (const point &amp;p) const</code></td>
<td>test if point is inside frustum</td>
</tr>
<tr>
<td><code>uint clipmask (const point &amp;p) const</code></td>
<td>get clip bitmask of point (0 if inside, (1&lt;&lt;PlaneIndex) if outside)</td>
</tr>
<tr>
<td><code>ClipStatus::Type clip (const line &amp;l, line &amp;clippedLine)</code></td>
<td>clip line against view frustum</td>
</tr>
<tr>
<td><code>ClipStatus::Type clipstatus (const bbox &amp;box)</code></td>
<td>get clip status of a local bounding box</td>
</tr>
<tr>
<td><code>ClipStatus::Type clipstatus (const bbox &amp;box, const matrix44 &amp;boxTransform)</code></td>
<td>get clip status of a transformed bounding box</td>
</tr>
<tr>
<td>template&lt;typename T&gt; <code>T as () const</code></td>
<td>convert to any type</td>
</tr>
</tbody>
</table>


void
Math::frustum::set(const matrix44 invViewProj) [inline]

setup from view and proj matrix

Setup frustum from invViewProj matrix (transform from projection space into world space).

void
Math::frustum::set(const bbox & box,
const matrix44 boxTransform &
) [inline]

setup from transformed bounding box

Setup from a transformed bounding box.

bool
Math::frustum::inside(const point & p) const [inline]

test if point is inside frustum

Test if point is inside frustum.

uint
Math::frustum::clipmask(const point & p) const [inline]

get clip bitmask of point (0 if inside, (1<<PlaneIndex) if outside)

Get clipmask of point.

ClipStatus::Type
Math::frustum::clipstatus(const bbox & box,
const matrix44 boxTransform)
get clip status of a transformed bounding box

Returns the clip status of a transformed bounding box.
Math::line Class Reference

#include <line.h>
Detailed Description

A line in 3d space.

(C) 2004 RadonLabs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>line ()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>line (const point &amp;startPoint, const point &amp;endPoint)</code></td>
<td>component constructor</td>
</tr>
<tr>
<td><code>line (const line &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>void set (const point &amp;startPoint, const point &amp;endPoint)</code></td>
<td>set start and end point</td>
</tr>
<tr>
<td><code>const point &amp; start () const</code></td>
<td>get start point</td>
</tr>
<tr>
<td><code>const point &amp; end () const</code></td>
<td>get end point</td>
</tr>
<tr>
<td><code>const vector &amp; vec () const</code></td>
<td>get vector</td>
</tr>
<tr>
<td><code>scalar length () const</code></td>
<td>get length</td>
</tr>
<tr>
<td><code>scalar lengthsq () const</code></td>
<td>get squared length</td>
</tr>
<tr>
<td><code>scalar distance (const point &amp;p) const</code></td>
<td>minimal distance of point to line</td>
</tr>
<tr>
<td><code>bool intersect (const line &amp;l, point &amp;pa, point &amp;pb) const</code></td>
<td>intersect with line</td>
</tr>
<tr>
<td><code>scalar closestpoint (const point &amp;p) const</code></td>
<td>return t of the closest point on the line</td>
</tr>
<tr>
<td><code>point pointat (scalar t) const</code></td>
<td>return ( p = b + mt )</td>
</tr>
</tbody>
</table>
### Member Function Documentation

#### bool Math::line::intersect(const line & l, point & pa, point & pb) const

**intersect with line**

Get line/line intersection. Returns the shortest line between two lines.

**Todo:**

: Untested! Replace with simpler code.

#### scalar Math::line::closestpoint(const point & p) const [inline]

**return** $t$ of the closest point on the line

Returns a point on the line which is closest to a another point in space. This just returns the parameter $t$ on where the point is located. If $t$ is between 0 and 1, the point is on the line, otherwise not. To get the actual 3d point $p$:

\[ p = m + b \cdot t \]

#### point Math::line::pointat(scalar t) const [inline]

**return** $p = b + m \cdot t$

Returns $p = b + m \cdot t$, given t. Note that the point is not on the line if $0.0 > t > 1.0$
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Math::matrix44
Math::matrix44 Class Reference

#include <xna_matrix44.h>
Detailed Description

A matrix44 class on top of Xbox360 math functions.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>matrix44()</code></td>
<td>default constructor, <em>NOTE:</em> does NOT setup components!</td>
</tr>
<tr>
<td><code>matrix44(float4 const &amp;row0, float4 const &amp;row1, float4 const &amp;row2, float4 const &amp;row3)</code></td>
<td>construct from components</td>
</tr>
<tr>
<td><code>matrix44(const XMMATRIX &amp;rhs)</code></td>
<td>construct from XMMATRIX</td>
</tr>
<tr>
<td><code>void operator=(const matrix44 &amp;rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td><code>void operator=(const XMMATRIX &amp;rhs)</code></td>
<td>assign XMMATRIX</td>
</tr>
<tr>
<td><code>bool operator==(const matrix44 &amp;rhs) const</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const matrix44 &amp;rhs) const</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>void load(const scalar *ptr)</code></td>
<td>load content from 16-byte-aligned memory</td>
</tr>
<tr>
<td><code>void loadu(const scalar *ptr)</code></td>
<td>load content from unaligned memory</td>
</tr>
<tr>
<td><code>void store(scalar *ptr) const</code></td>
<td>write content to 16-byte-aligned memory through the write cache</td>
</tr>
<tr>
<td><code>void storeu(scalar *ptr) const</code></td>
<td>write content to unaligned memory through the write cache</td>
</tr>
<tr>
<td><code>void stream(scalar *ptr) const</code></td>
<td>stream content to 16-byte-aligned memory circumventing the write-cache</td>
</tr>
<tr>
<td><code>void set(float4 const &amp;row0, float4 const &amp;row1, float4 const &amp;row2, float4 const &amp;row3)</code></td>
<td>set content</td>
</tr>
<tr>
<td><code>void setrow0(float4 const &amp;row0)</code></td>
<td>write access to x component</td>
</tr>
<tr>
<td><code>void setrow1(float4 const &amp;row1)</code></td>
<td>write access to y component</td>
</tr>
<tr>
<td><code>void setrow2(float4 const &amp;row2)</code></td>
<td></td>
</tr>
</tbody>
</table>
write access to z component

```cpp
void setrow3 (float4 const &row3)
```
write access to w component

```cpp
const float4 & getrow0 () const
```
read-only access to x component

```cpp
const float4 & getrow1 () const
```
read-only access to y component

```cpp
const float4 & getrow2 () const
```
read-only access to z component

```cpp
const float4 & getrow3 () const
```
read-only access to w component

```cpp
void set_xaxis (float4 const &x)
```
write access to x component

```cpp
void set_yaxis (float4 const &y)
```
write access to y component

```cpp
void set_zaxis (float4 const &z)
```
write access to z component

```cpp
void set_position (float4 const &pos)
```
write access to w component / pos component

```cpp
const float4 & get_xaxis () const
```
read access to x component

```cpp
const float4 & get_yaxis () const
```
read access to y component

```cpp
const float4 & get_zaxis () const
```
read access to z component

```cpp
const float4 & get_position () const
```
read access to w component / pos component

```cpp
void translate (float4 const &t)
```
add a translation to pos_component

```cpp
void scale (float4 const &v)
```
scale matrix

```cpp
bool isidentity () const
```
return true if matrix is identity

```cpp
scalar determinant () const
```
return determinant of matrix

```cpp
void decompose (float4 &outScale, quaternion
```
template<typename T>
    T as () const

convert to any type
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>identity ()</strong></td>
<td>build identity matrix</td>
</tr>
<tr>
<td><strong>affinetransformation</strong> (scalar scaling, float4 const &amp;rotationCenter, const quaternion &amp;rotation, float4 const &amp;translation)</td>
<td>build matrix from affine transformation</td>
</tr>
<tr>
<td><strong>inverse</strong> (const matrix44 &amp;m)</td>
<td>compute the inverse of a matrix</td>
</tr>
<tr>
<td><strong>lookatlh</strong> (float4 const &amp;eye, float4 const &amp;at, float4 const &amp;up)</td>
<td>build left handed lookat matrix</td>
</tr>
<tr>
<td><strong>lookatrh</strong> (float4 const &amp;eye, float4 const &amp;at, float4 const &amp;up)</td>
<td>build right handed lookat matrix</td>
</tr>
<tr>
<td><strong>multiply</strong> (const matrix44 &amp;m0, const matrix44 &amp;m1)</td>
<td>multiply 2 matrices</td>
</tr>
<tr>
<td><strong>ortholh</strong> (scalar w, scalar h, scalar zn, scalar zf)</td>
<td>build left handed orthogonal projection matrix</td>
</tr>
<tr>
<td><strong>orthorh</strong> (scalar w, scalar h, scalar zn, scalar zf)</td>
<td>build right handed orthogonal projection matrix</td>
</tr>
<tr>
<td><strong>orthooffcenterlh</strong> (scalar l, scalar r, scalar b, scalar t, scalar zn, scalar zf)</td>
<td>build left-handed off-center orthogonal projection matrix</td>
</tr>
<tr>
<td><strong>orthooффcenterrh</strong> (scalar l, scalar r, scalar b, scalar t, scalar zn, scalar zf)</td>
<td>build right-handed off-center orthogonal projection matrix</td>
</tr>
<tr>
<td><strong>perspfovlh</strong> (scalar fovy, scalar aspect, scalar zn, scalar zf)</td>
<td>build left-handed perspective projection matrix based on field-of-view</td>
</tr>
<tr>
<td><strong>perspfovrh</strong> (scalar fovy, scalar aspect, scalar zn, scalar zf)</td>
<td>build right-handed perspective projection matrix based on field-of-view</td>
</tr>
<tr>
<td><strong>persplh</strong> (scalar w, scalar h, scalar zn, scalar zf)</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>build left-handed perspective projection matrix</code></td>
<td></td>
</tr>
<tr>
<td><code>static matrix44 persprh (scalar w, scalar h, scalar zn, scalar zf)</code></td>
<td>build right-handed perspective projection matrix</td>
</tr>
<tr>
<td><code>static matrix44 perspoffcenterlh (scalar l, scalar r, scalar b, scalar t, scalar zn, scalar zf)</code></td>
<td>build left-handed off-center perspective projection matrix</td>
</tr>
<tr>
<td><code>static matrix44 perspoffcenterrh (scalar l, scalar r, scalar b, scalar t, scalar zn, scalar zf)</code></td>
<td>build right-handed off-center perspective projection matrix</td>
</tr>
<tr>
<td><code>static matrix44 reflect (const plane &amp;p)</code></td>
<td>build matrix that reflects coordinates about a plane</td>
</tr>
<tr>
<td><code>static matrix44 rotationaxis (float4 const &amp;axis, scalar angle)</code></td>
<td>build rotation matrix around arbitrary axis</td>
</tr>
<tr>
<td><code>static matrix44 rotationquaternion (const quaternion &amp;q)</code></td>
<td>build rotation matrix from quaternion</td>
</tr>
<tr>
<td><code>static matrix44 rotationx (scalar angle)</code></td>
<td>build x-axis-rotation matrix</td>
</tr>
<tr>
<td><code>static matrix44 rotationy (scalar angle)</code></td>
<td>build y-axis-rotation matrix</td>
</tr>
<tr>
<td><code>static matrix44 rotationz (scalar angle)</code></td>
<td>build z-axis-rotation matrix</td>
</tr>
<tr>
<td><code>static matrix44 rotationyawpitchroll (scalar yaw, scalar pitch, scalar roll)</code></td>
<td>build rotation matrix from yaw, pitch and roll</td>
</tr>
<tr>
<td><code>static matrix44 scaling (scalar sx, scalar sy, scalar sz)</code></td>
<td>build a scaling matrix from components</td>
</tr>
<tr>
<td><code>static matrix44 scaling (float4 const &amp;s)</code></td>
<td>build a scaling matrix from float4</td>
</tr>
<tr>
<td><code>static matrix44 transformation (float4 const &amp;scalingCenter, const quaternion &amp;scalingRotation, float4 const &amp;scaling, float4 const &amp;rotationCenter, const quaternion &amp;rotation, float4 const &amp;translation)</code></td>
<td>build a transformation matrix</td>
</tr>
<tr>
<td><code>static matrix44 translation (scalar x, scalar y, scalar z)</code></td>
<td>build a translation matrix</td>
</tr>
<tr>
<td><code>static matrix44 translation (float4 const &amp;t)</code></td>
<td>build a translation matrix from point</td>
</tr>
<tr>
<td>Function Type</td>
<td>Function</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>static matrix44</td>
<td>transpose (const matrix44 &amp;m)</td>
</tr>
<tr>
<td>static float4</td>
<td>transform (const float4 &amp;v, const matrix44 &amp;m)</td>
</tr>
<tr>
<td>static quaternion</td>
<td>rotationmatrix (const matrix44 &amp;m)</td>
</tr>
<tr>
<td>static plane</td>
<td>transform (const plane &amp;p, const matrix44 &amp;m)</td>
</tr>
<tr>
<td>static bool</td>
<td>ispointinside (const float4 &amp;p, const matrix44 &amp;m)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Math::matrix44::decompose(
    float4 & outScale,
    quaternion & outRotation,
    float4 & outTranslation
) const

decompose into scale, rotation and translation !!! Note:
```

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:47 2010
Math::noise
Math::noise Class Reference

#include <noise.h>
Detailed Description

Perlin noise class.

See http://mrl.nyu.edu/~perlin/noise/ for details.

(C) 2006 RadonLabs GmbH
Static Public Member Functions

static float gen (float x, float y, float z)
generate noise value
Math::plane
Math::plane Class Reference

#include <xna_plane.h>
Detailed Description

Nebula's plane class.

(C) 2007 RadonLabs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>plane ()</code></td>
<td>default constructor, NOTE: does NOT setup components!</td>
</tr>
<tr>
<td><code>plane (scalar a, scalar b, scalar c, scalar d)</code></td>
<td>construct from components</td>
</tr>
<tr>
<td><code>plane (const float4 &amp;p0, const float4 &amp;p1, const float4 &amp;p2)</code></td>
<td>construct from points</td>
</tr>
<tr>
<td><code>plane (const float4 &amp;p, const float4 &amp;n)</code></td>
<td>construct from point and normal</td>
</tr>
<tr>
<td><code>plane (XMVECTOR rhs)</code></td>
<td>construct from XMVECTOR</td>
</tr>
<tr>
<td><code>void setup_from_points (const float4 &amp;p0, const float4 &amp;p1, const float4 &amp;p2)</code></td>
<td>setup from points</td>
</tr>
<tr>
<td><code>void setup_from_point_and_normal (const float4 &amp;p, const float4 &amp;n)</code></td>
<td>setup from point and normal</td>
</tr>
<tr>
<td><code>void set (scalar a, scalar b, scalar c, scalar d)</code></td>
<td>set components</td>
</tr>
<tr>
<td><code>void set_a (scalar a)</code></td>
<td>set the x component</td>
</tr>
<tr>
<td><code>void set_b (scalar b)</code></td>
<td>set the y component</td>
</tr>
<tr>
<td><code>void set_c (scalar c)</code></td>
<td>set the z component</td>
</tr>
<tr>
<td><code>void set_d (scalar d)</code></td>
<td>set the w component</td>
</tr>
<tr>
<td><code>scalar &amp; a ()</code></td>
<td>read/write access to A component</td>
</tr>
<tr>
<td><code>scalar &amp; b ()</code></td>
<td>read/write access to B component</td>
</tr>
<tr>
<td><code>scalar &amp; c ()</code></td>
<td>read/write access to C component</td>
</tr>
</tbody>
</table>
**scalar** & **d** ()
*read/write access to D component*

**scalar** **a** () **const**
*read-only access to A component*

**scalar** **b** () **const**
*read-only access to B component*

**scalar** **c** () **const**
*read-only access to C component*

**scalar** **d** () **const**
*read-only access to D component*

**scalar** **dot** (const **float4** &v) **const**
*compute dot product of plane and vector*

**bool** **intersectLine** (const **float4** &startPoint, const **float4** &endPoint, **float4** &outIntersectPoint) **const**
*find intersection with line*

**ClipStatus::Type** **clip** (const **line** &l, **line** &outClippedLine) **const**
*clip line against this plane*
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static plane</th>
<th><strong>normalize</strong> (const plane &amp;p)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>normalize plane components a, b, c</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static __declspec (deprecated) plane</th>
<th><strong>transform</strong>(PlaneArg p)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>transform plane by inverse transpose of transform</td>
</tr>
</tbody>
</table>
Constructor & Destructor Documentation

Math::plane::plane ( scalar  a,
                  scalar  b,
                  scalar  c,
                  scalar  d
               ) [inline]

construct from components

!!!! copy constructor forbidden, otherwise passing plane's to a function
!!!! via Registers doesnt work
Math::point
Math::point Class Reference

#include <xna_point.h>

Inheritance diagram for Math::point:

```
Math::float4
  ^
  | Math::point
```
Detailed Description

A point in homogenous space. A point describes a position in space, and has its W component set to 1.0.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>point ()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>point (scalar x, scalar y, scalar z)</code></td>
<td>construct from components</td>
</tr>
<tr>
<td><code>point (const float4 &amp;rhs)</code></td>
<td>construct from <code>float4</code></td>
</tr>
<tr>
<td><code>point (XMVECTOR rhs)</code></td>
<td>construct from <code>XMVECTOR</code></td>
</tr>
<tr>
<td><code>void operator= (const point &amp;rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td><code>void operator= (XMVECTOR rhs)</code></td>
<td>assign <code>XMVECTOR</code></td>
</tr>
<tr>
<td><code>void operator+= (const vector &amp;rhs)</code></td>
<td>inplace add vector</td>
</tr>
<tr>
<td><code>void operator-= (const vector &amp;rhs)</code></td>
<td>inplace subtract vector</td>
</tr>
<tr>
<td><code>point operator+ (const vector &amp;rhs)</code> const</td>
<td>add point and vector</td>
</tr>
<tr>
<td><code>point operator- (const vector &amp;rhs)</code> const</td>
<td>subtract vectors from point</td>
</tr>
<tr>
<td><code>vector operator- (const point &amp;rhs)</code> const</td>
<td>subtract point from point into a vector</td>
</tr>
<tr>
<td><code>bool operator== (const point &amp;rhs)</code> const</td>
<td>equality operator</td>
</tr>
<tr>
<td><code>bool operator!= (const point &amp;rhs)</code> const</td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>void set (scalar x, scalar y, scalar z)</code></td>
<td>set components</td>
</tr>
<tr>
<td><code>float4 operator- ()</code> const</td>
<td>flip sign</td>
</tr>
<tr>
<td><code>float4 operator- (const float4 &amp;rhs)</code> const</td>
<td>subtract 2 vectors</td>
</tr>
<tr>
<td><code>void operator+= (const float4 &amp;rhs)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>inplace add</td>
<td>In-place addition of two vectors.</td>
</tr>
<tr>
<td>void operator-=(const float4 &amp;rhs)</td>
<td>Subtract rhs from this vector.</td>
</tr>
<tr>
<td>inplace sub</td>
<td>In-place scalar multiply.</td>
</tr>
<tr>
<td>void operator *(scalar s)</td>
<td>Multiply this vector by scalar.</td>
</tr>
<tr>
<td>void operator *= (const float4 &amp;rhs)</td>
<td>Multiply this vector by vector component-wise.</td>
</tr>
<tr>
<td>float4 operator+ (const float4 &amp;rhs) const</td>
<td>Add two vectors.</td>
</tr>
<tr>
<td>float4 operator * (scalar s) const</td>
<td>Multiply this vector by scalar.</td>
</tr>
<tr>
<td>bool operator== (const float4 &amp;rhs) const</td>
<td>Equality operator.</td>
</tr>
<tr>
<td>bool operator!= (const float4 &amp;rhs) const</td>
<td>Inequality operator.</td>
</tr>
<tr>
<td>void load (const scalar *ptr)</td>
<td>Load content from 16-byte-aligned memory.</td>
</tr>
<tr>
<td>void loadu (const scalar *ptr)</td>
<td>Load content from unaligned memory.</td>
</tr>
<tr>
<td>void store (scalar *ptr) const</td>
<td>Write content to 16-byte-aligned memory through the write cache.</td>
</tr>
<tr>
<td>void storeu (scalar *ptr) const</td>
<td>Write content to unaligned memory through the write cache.</td>
</tr>
<tr>
<td>void stream (scalar *ptr) const</td>
<td>Stream content to 16-byte-aligned memory circumventing the write-cache.</td>
</tr>
<tr>
<td>void load_float3 (const void *ptr, float w)</td>
<td>Load 3 floats into x,y,z from unaligned memory.</td>
</tr>
<tr>
<td>void load_ubyte4n_signed (const void *ptr, float w)</td>
<td>Load from UByte4N packed vector, move range to -1..+1.</td>
</tr>
<tr>
<td>void set (scalar x, scalar y, scalar z, scalar w)</td>
<td>Set content.</td>
</tr>
<tr>
<td>void set_x (scalar x)</td>
<td>Set the x component.</td>
</tr>
<tr>
<td>void set_y (scalar y)</td>
<td>Set the y component.</td>
</tr>
<tr>
<td>void set_z (scalar z)</td>
<td>Set the z component.</td>
</tr>
<tr>
<td>void</td>
<td><strong>set_w</strong> (scalar w)</td>
</tr>
<tr>
<td>------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td>set the w component</td>
</tr>
<tr>
<td>scalar &amp;</td>
<td><strong>x</strong> ()</td>
</tr>
<tr>
<td></td>
<td>read/write access to x component</td>
</tr>
<tr>
<td>scalar</td>
<td><strong>x</strong> () const</td>
</tr>
<tr>
<td></td>
<td>read-only access to x component</td>
</tr>
<tr>
<td>scalar &amp;</td>
<td><strong>y</strong> ()</td>
</tr>
<tr>
<td></td>
<td>read/write access to y component</td>
</tr>
<tr>
<td>scalar</td>
<td><strong>y</strong> () const</td>
</tr>
<tr>
<td></td>
<td>read-only access to y component</td>
</tr>
<tr>
<td>scalar &amp;</td>
<td><strong>z</strong> ()</td>
</tr>
<tr>
<td></td>
<td>read/write access to z component</td>
</tr>
<tr>
<td>scalar</td>
<td><strong>z</strong> () const</td>
</tr>
<tr>
<td></td>
<td>read-only access to z component</td>
</tr>
<tr>
<td>scalar &amp;</td>
<td><strong>w</strong> ()</td>
</tr>
<tr>
<td></td>
<td>read/write access to w component</td>
</tr>
<tr>
<td>scalar</td>
<td><strong>w</strong> () const</td>
</tr>
<tr>
<td></td>
<td>read-only access to w component</td>
</tr>
<tr>
<td>scalar</td>
<td><strong>length</strong> () const</td>
</tr>
<tr>
<td></td>
<td>return length of vector</td>
</tr>
<tr>
<td>scalar</td>
<td><strong>lengthsq</strong> () const</td>
</tr>
<tr>
<td></td>
<td>return squared length of vector</td>
</tr>
<tr>
<td><strong>float4</strong></td>
<td><strong>abs</strong> () const</td>
</tr>
<tr>
<td></td>
<td>return component-wise absolute</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>origin()</code></td>
<td>return a point at the origin (0, 0, 0)</td>
</tr>
<tr>
<td><code>reciprocal()</code></td>
<td>return 1.0 / vec</td>
</tr>
<tr>
<td><code>multiply()</code></td>
<td>component-wise multiplication</td>
</tr>
<tr>
<td><code>cross3()</code></td>
<td>return 3-dimensional cross product</td>
</tr>
<tr>
<td><code>dot3()</code></td>
<td>return 3d dot product of vectors</td>
</tr>
<tr>
<td><code>barycentric()</code></td>
<td>return point in barycentric coordinates</td>
</tr>
<tr>
<td><code>catmullrom()</code></td>
<td>perform Catmull-Rom interpolation</td>
</tr>
<tr>
<td><code>hermite()</code></td>
<td>perform Hermite spline interpolation</td>
</tr>
<tr>
<td><code>lerp()</code></td>
<td>perform linear interpolation between 2 4d vectors</td>
</tr>
<tr>
<td><code>maximize()</code></td>
<td>return 4d vector made up of largest components of 2 vectors</td>
</tr>
<tr>
<td><code>minimize()</code></td>
<td>return 4d vector made up of smallest components of 2 vectors</td>
</tr>
<tr>
<td><code>normalize()</code></td>
<td>return normalized version of 4d vector</td>
</tr>
<tr>
<td><code>transform()</code></td>
<td>transform 4d vector by <code>matrix44</code></td>
</tr>
<tr>
<td><code>reflect()</code></td>
<td>reflect a vector</td>
</tr>
<tr>
<td><code>clamp()</code></td>
<td>reflect a vector</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>angle</code></td>
<td>angle between two vectors</td>
</tr>
<tr>
<td><code>less3_any</code></td>
<td>return true if any XYZ component is less-than</td>
</tr>
<tr>
<td><code>less3_all</code></td>
<td>return true if all XYZ components are less-than</td>
</tr>
<tr>
<td><code>lessequal3_any</code></td>
<td>return true if any XYZ component is less-or-equal</td>
</tr>
<tr>
<td><code>lessequal3_all</code></td>
<td>return true if all XYZ components are less-or-equal</td>
</tr>
<tr>
<td><code>greater3_any</code></td>
<td>return true if any XYZ component is greater</td>
</tr>
<tr>
<td><code>greater3_all</code></td>
<td>return true if all XYZ components are greater</td>
</tr>
<tr>
<td><code>greaterequal3_any</code></td>
<td>return true if any XYZ component is greater-or-equal</td>
</tr>
<tr>
<td><code>greaterequal3_all</code></td>
<td>return true if all XYZ components are greater-or-equal</td>
</tr>
<tr>
<td><code>equal3_any</code></td>
<td>return true if any XYZ component is equal</td>
</tr>
<tr>
<td><code>equal3_all</code></td>
<td>return true if all XYZ components are equal</td>
</tr>
<tr>
<td><code>nearequal3</code></td>
<td>perform near equal comparison with given epsilon (3 components)</td>
</tr>
<tr>
<td><code>less4_any</code></td>
<td>return true if any XZW component is less-than</td>
</tr>
<tr>
<td><code>less4_all</code></td>
<td>return true if all XZW components are less-than</td>
</tr>
<tr>
<td><code>lessequal4_any</code></td>
<td>return true if any XZW component is less-or-equal</td>
</tr>
<tr>
<td><code>lessequal4_all</code></td>
<td>return true if all XZW components are less-or-equal</td>
</tr>
<tr>
<td><code>greater4_any</code></td>
<td>return true if any XZW component is greater</td>
</tr>
</tbody>
</table>
static bool greater4_all (const float4 &v0, const float4 &v1)  
return true if all XYZW components are greater

static bool greaterequal4_any (const float4 &v0, const float4 &v1)  
return true if any XYZW component is greater-or-equal

static bool greaterequal4_all (const float4 &v0, const float4 &v1)  
return true if all XYZW components are greater-or-equal

static bool equal4_any (const float4 &v0, const float4 &v1)  
return true if any XYZW component is equal

static bool equal4_all (const float4 &v0, const float4 &v1)  
return true if all XYZW components are equal

static bool nearequal4 (const float4 &v0, const float4 &v1, const float4 &epsilon)  
perform near equal comparison with given epsilon (4 components)

static float unpack_x (XMVECTOR v)  
unpack the first element from a XMVECTOR

static float unpack_y (XMVECTOR v)  
unpack the second element from a XMVECTOR

static float unpack_z (XMVECTOR v)  
unpack the third element from a XMVECTOR

static float unpack_w (XMVECTOR v)  
unpack the fourth element from a XMVECTOR

static float4 splat (scalar s)  
splat scalar into each component of a vector

static float4 splat (const float4 &v, uint element)  
return a vector with all elements set to element n of v. 0 <= element <= 3

static float4 splat_x (const float4 &v)  
return a vector with all elements set to v.x

static float4 splat_y (const float4 &v)  
return a vector with all elements set to v.y

static float4 splat_z (const float4 &v)  
return a vector with all elements set to v.z

static float4 splat_w (const float4 &v)  
return a vector with all elements set to v.w

static float4 permute_control (unsigned int i0, unsigned int i1, unsigned int i2, unsigned int i3)  
return control vector for permute (see XMVectorPermuteControl for details)
static `float4` **permute** (const `float4` &v0, const `float4` &v1, const `float4` &control)

merge components of 2 vectors into 1 (see `XMVectorPermute` for details)
Constructor & Destructor Documentation

```cpp
__forceinline
Math::point::point(XMVECTOR rhs)
```

construct from XMVECTOR

!!!! copy constructor forbidden, otherwise passing point's to a function
!!!! via Registers doesn't work
Member Function Documentation

```cpp
__forceinline
void
Math::float4::load
  ( scalar ptr ) [inherited]

load content from 16-byte-aligned memory
Load 4 floats from 16-byte-aligned memory.
```

```cpp
__forceinline void
Math::float4::loadu
  ( scalar ptr ) [inherited]

load content from unaligned memory
Load 4 floats from unaligned memory.
```

```cpp
__forceinline void
Math::float4::store
  ( * ptr ) const [inherited]

write content to 16-byte-aligned memory through the write cache
Store to 16-byte-aligned float pointer.
```

```cpp
__forceinline void
Math::float4::storeu
  ( * ptr ) const [inherited]

write content to unaligned memory through the write cache
Store to non-aligned float pointer.
```
Math::polar
Math::polar Class Reference

#include <polar.h>
Detailed Description

A polar coordinate inline class, consisting of 2 angles theta (latitude) and rho (longitude). Also offers conversion between cartesian and polar space.

Allowed range for theta is 0..180 degree (in rad!) and for rho 0..360 degree (in rad).

(C) 2004 RadonLabs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>polar ()</strong></td>
<td>the default constructor</td>
</tr>
<tr>
<td><strong>polar (scalar t, scalar r)</strong></td>
<td>constructor, theta and rho args</td>
</tr>
<tr>
<td><strong>polar (const vector &amp;v)</strong></td>
<td>constructor, normalized cartesian vector as arg</td>
</tr>
<tr>
<td><strong>polar (const polar &amp;src)</strong></td>
<td>the copy constructor</td>
</tr>
<tr>
<td><strong>void operator= (const polar &amp;rhs)</strong></td>
<td>the assignment operator</td>
</tr>
<tr>
<td><strong>vector get_cartesian () const</strong></td>
<td>convert to normalized cartesian coords</td>
</tr>
<tr>
<td><strong>void set (const polar &amp;p)</strong></td>
<td>set to polar object</td>
</tr>
<tr>
<td><strong>void set (scalar t, scalar r)</strong></td>
<td>set to theta and rho</td>
</tr>
<tr>
<td><strong>void set (const vector &amp;v)</strong></td>
<td>set to cartesian</td>
</tr>
</tbody>
</table>
Member Function Documentation

vector
Math::polar::get_cartesian ( ) const [inline]

convert to normalized cartesian coords

Convert polar to cartesian.

void
Math::polar::set ( vector vec ) [inline]

set to cartesian

Convert cartesian to polar.
Math::quaternion
Math::quaternion Class Reference

#include <xna_quaternion.h>
Detailed Description

Nebula's quaternion class.

(C) 2004 RadonLabs GmbH
### Public Member Functions

- **quaternion ()**
  
  *default constructor, NOTE: does NOT setup components!*

- **quaternion (scalar x, scalar y, scalar z, scalar w)**
  
  *construct from components*

- **quaternion (float4 const &rhs)**
  
  *construct from float4*

- **quaternion (XMVECTOR rhs)**
  
  *construct from XMVECTOR*

- **void operator= (const quaternion &rhs)**
  
  *assignment operator*

- **void operator= (XMVECTOR rhs)**
  
  *assign XMVECTOR*

- **bool operator== (const quaternion &rhs) const**
  
  *equality operator*

- **bool operator!= (const quaternion &rhs) const**
  
  *inequality operator*

- **void load (const scalar *ptr)**
  
  *load content from 16-byte-aligned memory*

- **void loadu (const scalar *ptr)**
  
  *load content from unaligned memory*

- **void store (scalar *ptr) const**
  
  *write content to 16-byte-aligned memory through the write cache*

- **void storeu (scalar *ptr) const**
  
  *write content to unaligned memory through the write cache*

- **void stream (scalar *ptr) const**
  
  *stream content to 16-byte-aligned memory circumventing the write-cache*

- **void set (scalar x, scalar y, scalar z, scalar w)**
  
  *set content*

- **void set (float4 const &f4)**
  
  *set from float4*

- **void set_x (scalar x)**
  
  *set the x component*

- **void set_y (scalar y)**
set the \textit{y} component

\textbf{void} \texttt{set\_z} (scalar \texttt{z})
set the \textit{z} component

\textbf{void} \texttt{set\_w} (scalar \texttt{w})
set the \textit{w} component

scalar \& \texttt{x} ()
read/write access to \textit{x} component

scalar \& \texttt{y} ()
read/write access to \textit{y} component

scalar \& \texttt{z} ()
read/write access to \textit{z} component

scalar \& \texttt{w} ()
read/write access to \textit{w} component

scalar \texttt{x} () \texttt{const}
read-only access to \textit{x} component

scalar \texttt{y} () \texttt{const}
read-only access to \textit{y} component

scalar \texttt{z} () \texttt{const}
read-only access to \textit{z} component

scalar \texttt{w} () \texttt{const}
read-only access to \textit{w} component

\textbf{bool} \texttt{isidentity} () \texttt{const}
return true if quaternion is identity

scalar \texttt{length} () \texttt{const}
returns length

scalar \texttt{lengthsq} () \texttt{const}
returns length squared

\textbf{void} \texttt{undenormalize} ()
un-denormalize quaternion (this is sort of a hack since Maya likes to return
denormal quaternions)
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>barycentric</code> (const <code>quaternion</code> &amp;q0, const <code>quaternion</code> &amp;q1, const <code>quaternion</code> &amp;q2, scalar f, scalar g)</td>
<td>Return quaternion in barycentric coordinates</td>
</tr>
<tr>
<td><code>conjugate</code> (const <code>quaternion</code> &amp;q)</td>
<td>Return conjugate of a normalized quaternion</td>
</tr>
<tr>
<td><code>dot</code> (const <code>quaternion</code> &amp;q0, const <code>quaternion</code> &amp;q1)</td>
<td>Return dot product of two normalized quaternions</td>
</tr>
<tr>
<td><code>exp</code> (const <code>quaternion</code> &amp;q0)</td>
<td>Calculate the exponential</td>
</tr>
<tr>
<td><code>identity</code> ()</td>
<td>Returns an identity quaternion</td>
</tr>
<tr>
<td><code>inverse</code> (const <code>quaternion</code> &amp;q)</td>
<td>Conjugates and renormalizes quaternion</td>
</tr>
<tr>
<td><code>In</code> (const <code>quaternion</code> &amp;q)</td>
<td>Calculate the natural logarithm</td>
</tr>
<tr>
<td><code>multiply</code> (const <code>quaternion</code> &amp;q0, const <code>quaternion</code> &amp;q1)</td>
<td>Multiply 2 quaternions</td>
</tr>
<tr>
<td><code>normalize</code> (const <code>quaternion</code> &amp;q)</td>
<td>Compute unit length quaternion</td>
</tr>
<tr>
<td><code>rotationaxis</code> (const <code>float4</code> &amp;axis, scalar angle)</td>
<td>Build quaternion from axis and clockwise rotation angle in radians</td>
</tr>
<tr>
<td><code>rotationmatrix</code> (const <code>matrix44</code> &amp;m)</td>
<td>Build quaternion from rotation matrix</td>
</tr>
<tr>
<td><code>rotationyawpitchroll</code> (scalar yaw, scalar pitch, scalar roll)</td>
<td>Build quaternion from yaw, pitch and roll</td>
</tr>
<tr>
<td><code>slerp</code> (const <code>quaternion</code> &amp;q1, const <code>quaternion</code> &amp;q2, scalar t)</td>
<td>Interpolate between 2 quaternion using spherical interpolation</td>
</tr>
<tr>
<td><code>squadsetup</code> (const <code>quaternion</code> &amp;q0, const <code>quaternion</code> &amp;q1, const <code>quaternion</code> &amp;q2, const scalar t)</td>
<td>Build quaternion from yaw, pitch and roll with interpolation parameters</td>
</tr>
<tr>
<td><strong>quaternion</strong> &amp;q3, <strong>quaternion</strong> &amp;aOut, <strong>quaternion</strong> &amp;bOut, <strong>quaternion</strong> &amp;cOut</td>
<td></td>
</tr>
<tr>
<td>setup control points for spherical quadrangle interpolation</td>
<td></td>
</tr>
<tr>
<td><strong>static</strong> <strong>quaternion</strong> squad (const <strong>quaternion</strong> &amp;q1, const <strong>quaternion</strong> &amp;a, const <strong>quaternion</strong> &amp;b, const <strong>quaternion</strong> &amp;c, scalar t)</td>
<td></td>
</tr>
<tr>
<td>interpolate between quaternions using spherical quadrangle interpolation</td>
<td></td>
</tr>
<tr>
<td><strong>static void</strong> to_axisangle (const <strong>quaternion</strong> &amp;q, float4 &amp;outAxis, scalar &amp;outAngle)</td>
<td></td>
</tr>
<tr>
<td>convert quaternion to axis and angle</td>
<td></td>
</tr>
</tbody>
</table>
Constructor & Destructor Documentation

__forceinline

Math::quaternion::quaternion(XMVECTOR rhs)

construct from XMVECTOR

copy constructor !!!! copy constructor forbidden, otherwise passing point’s to a function !!!! via Registers doesn’t work
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**Math::rectangle**
Math::rectangle< TYPE > Class
Template Reference

#include <rectangle.h>
Detailed Description

`template<class TYPE>
class Math::rectangle< TYPE >`

A 2d rectangle class.

(C) 2003 RadonLabs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>rectangle ()</strong></td>
<td>default constructor</td>
</tr>
<tr>
<td><strong>rectangle (TYPE l, TYPE t, TYPE r, TYPE b)</strong></td>
<td>constructor 1</td>
</tr>
<tr>
<td><strong>set (TYPE l, TYPE t, TYPE r, TYPE b)</strong></td>
<td>set content</td>
</tr>
<tr>
<td><strong>inside (TYPE x, TYPE y) const</strong></td>
<td>return true if point is inside</td>
</tr>
<tr>
<td><strong>width () const</strong></td>
<td>return width</td>
</tr>
<tr>
<td><strong>height () const</strong></td>
<td>return height</td>
</tr>
<tr>
<td><strong>centerX () const</strong></td>
<td>return center x</td>
</tr>
<tr>
<td><strong>centerY () const</strong></td>
<td>return center y</td>
</tr>
</tbody>
</table>
Math::sphere
Math::sphere Class Reference

#include <sphere_.h>
Detailed Description

A 3-dimensional sphere.

(C) 2004 RadonLabs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sphere ()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>sphere (const point &amp;_p, scalar _r)</code></td>
<td>pos/radius constructor</td>
</tr>
<tr>
<td><code>sphere (scalar _x, scalar _y, scalar _z, scalar _r)</code></td>
<td>x,y,z,r constructor</td>
</tr>
<tr>
<td><code>sphere (const sphere &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>void set (const point &amp;_p, scalar _r)</code></td>
<td>set position and radius</td>
</tr>
<tr>
<td><code>void set (scalar _x, scalar _y, scalar _z, scalar _r)</code></td>
<td>set x,y,z, radius</td>
</tr>
<tr>
<td><code>bool inside (const bbox &amp;box) const</code></td>
<td>return true if box is completely inside sphere</td>
</tr>
<tr>
<td><code>bool intersects (const sphere &amp;s) const</code></td>
<td>check if 2 spheres overlap</td>
</tr>
<tr>
<td><code>bool intersects (const bbox &amp;box) const</code></td>
<td>check if sphere intersects box</td>
</tr>
<tr>
<td><code>bool intersect_sweep (const vector &amp;va, const sphere &amp;sb, const vector &amp;vb, scalar &amp;u0, scalar &amp;u1) const</code></td>
<td>check if 2 moving sphere have contact</td>
</tr>
<tr>
<td><code>project_screen_rh (const matrix44 &amp;modelView, const matrix44 &amp;projection, scalar nearZ) const</code></td>
<td>project sphere to screen rectangle (right handed coordinate system)</td>
</tr>
<tr>
<td><code>ClipStatus::Type clipstatus (const bbox &amp;box) const</code></td>
<td>get clip status of box against sphere</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
bool Math::sphere::inside (bbox box ) const [inline]

return true if box is completely inside sphere

Return true if the bounding box is inside the sphere.

bool Math::sphere::intersects (bbox box ) const

check if sphere intersects box

Check if sphere intersects with box. Taken from "Simple Intersection Tests For Games", Gamasutra, Oct 18 1999

bool Math::sphere::intersect_sweep (vector va,
       const sphere sb,
       const vector vb,
       scalar u0,
       scalar u1)

check if 2 moving sphere have contact

Check if 2 moving spheres have contact. Taken from "Simple Intersection Tests For Games" article in Gamasutra, Oct 18 1999

**Parameters:**

- `va` [in] distance travelled by 'this'
- `sb` [in] the other sphere
\( vb \) [in] distance travelled by sb
\( u0 \) [out] normalized intro contact
\( u1 \) [out] normalized outro contact

\texttt{rectangle< scalar > Math::sphere::project_screen_rh (}\texttt{const matrix44 view},
\texttt{&}
\texttt{const matrix44 projection,}
\texttt{&}
\texttt{scalar nearZ }) \texttt{const}

project sphere to screen rectangle (right handed coordinate system)

Project the sphere (defined in global space) to a screen space rectangle, given the current View and Projection matrices. The method assumes that the sphere is at least partially visible.

\texttt{ClipStatus::Type Math::sphere::clipstatus (bbox box ) const [inline]}

get clip status of box against sphere

Get the clip status of a box against this sphere. Inside means: the box is completely inside the sphere.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Math::transform44
Math::transform44 Class Reference

#include <transform44.h>
Detailed Description

A 4x4 matrix which is described by translation, rotation and scale.

(C) 2004 RadonLabs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>transform44 ()</code></td>
<td><strong>constructor</strong></td>
</tr>
<tr>
<td><code>void setposition (const point &amp;p)</code></td>
<td><strong>set position</strong></td>
</tr>
<tr>
<td><code>const point &amp; getposition () const</code></td>
<td><strong>get position</strong></td>
</tr>
<tr>
<td><code>void setrotate (const quaternion &amp;r)</code></td>
<td><strong>set rotate</strong></td>
</tr>
<tr>
<td><code>const quaternion &amp; getrotate () const</code></td>
<td><strong>get rotate</strong></td>
</tr>
<tr>
<td><code>void setscale (const vector &amp;s)</code></td>
<td><strong>set scale</strong></td>
</tr>
<tr>
<td><code>const vector &amp; getscale () const</code></td>
<td><strong>get scale</strong></td>
</tr>
<tr>
<td><code>void setrotatepivot (const point &amp;p)</code></td>
<td><strong>set optional rotate pivot</strong></td>
</tr>
<tr>
<td><code>const point &amp; getrotatepivot () const</code></td>
<td><strong>get optional rotate pivot</strong></td>
</tr>
<tr>
<td><code>void setscalepivot (const point &amp;p)</code></td>
<td><strong>set optional scale pivot</strong></td>
</tr>
<tr>
<td><code>const point &amp; getscalepivot () const</code></td>
<td><strong>get optional scale pivot</strong></td>
</tr>
<tr>
<td><code>void setoffset (const matrix44 &amp;m)</code></td>
<td><strong>set optional offset matrix</strong></td>
</tr>
<tr>
<td><code>const matrix44 &amp; getoffset () const</code></td>
<td><strong>get optional offset matrix</strong></td>
</tr>
<tr>
<td><code>const matrix44 &amp; getmatrix ()</code></td>
<td><strong>get resulting 4x4 matrix</strong></td>
</tr>
<tr>
<td><code>bool isdirty () const</code></td>
<td><strong>return true if the transformation matrix is dirty</strong></td>
</tr>
</tbody>
</table>
Math::vector
Math::vector Class Reference

#include <xna_vector.h>

Inheritance diagram for Math::vector:

```
Math::float4

Math::vector
```


Detailed Description

A vector in homogenous space. A point describes a direction and length in 3d space and always has a w component of 0.0.

(C) 2007 Radon Labs GmbH
Public Member Functions

```plaintext
vector ()
default constructor

vector (scalar x, scalar y, scalar z)
construct from components

vector (const float4 &rhs)
construct from float4

vector (XMVECTOR rhs)
construct from XMVECTOR

void operator= (const vector &rhs)
assignment operator

void operator= (XMVECTOR rhs)
assign XMVECTOR

vector operator- () const
flip sign

void operator+= (const vector &rhs)
add vector inplace

void operator-= (const vector &rhs)
subtract vector inplace

void operator*= (scalar s)
scale vector inplace

vector operator+ (const vector &rhs) const
add 2 vectors

vector operator- (const vector &rhs) const
subtract 2 vectors

vector operator* (scalar s) const
scale vector

bool operator== (const vector &rhs) const
equality operator

bool operator!= (const vector &rhs) const
inequality operator

void set (scalar x, scalar y, scalar z)
set components

float4 operator- (const float4 &rhs) const
```
subtract 2 vectors

```cpp
void operator+= (const float4 &rhs)
inplace add
```

```cpp
void operator-= (const float4 &rhs)
inplace sub
```

```cpp
void operator *= (const float4 &rhs)
multiply by a vector component-wise
```

```cpp
float4 operator+ (const float4 &rhs) const
add 2 vectors
```}

```cpp
bool operator== (const float4 &rhs) const
equality operator
```

```cpp
bool operator!= (const float4 &rhs) const
inequality operator
```

```cpp
void load (const scalar *ptr)
load content from 16-byte-aligned memory
```

```cpp
void loadu (const scalar *ptr)
load content from unaligned memory
```

```cpp
void store (scalar *ptr) const
write content to 16-byte-aligned memory through the write cache
```

```cpp
void storeu (scalar *ptr) const
write content to unaligned memory through the write cache
```

```cpp
void stream (scalar *ptr) const
stream content to 16-byte-aligned memory circumventing the write-cache
```

```cpp
void load_float3 (const void *ptr, float w)
load 3 floats into x,y,z from unaligned memory
```

```cpp
void load_ubyte4n_signed (const void *ptr, float w)
load from UByte4N packed vector, move range to -1..+1
```

```cpp
void set (scalar x, scalar y, scalar z, scalar w)
set content
```

```cpp
void set_x (scalar x)
set the x component
```

```cpp
void set_y (scalar y)
set the y component
```

```cpp
void set_z (scalar z)
set the z component
```

```cpp
void set_w (scalar w)
set the w component
```
scalar & \textbf{x} ()
\textit{read/write access to x component}

scalar \textbf{x} () const
\textit{read-only access to x component}

scalar & \textbf{y} ()
\textit{read/write access to y component}

scalar \textbf{y} () const
\textit{read-only access to y component}

scalar & \textbf{z} ()
\textit{read/write access to z component}

scalar \textbf{z} () const
\textit{read-only access to z component}

scalar & \textbf{w} ()
\textit{read/write access to w component}

scalar \textbf{w} () const
\textit{read-only access to w component}

scalar \textbf{length} () const
\textit{return length of vector}

scalar \textbf{lengthsq} () const
\textit{return squared length of vector}

\textbf{float4} \textbf{abs} () const
\textit{return component-wise absolute}
<table>
<thead>
<tr>
<th>Static Public Member Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static vector</strong> <strong>nullvec</strong> ()</td>
</tr>
<tr>
<td>return the null vector</td>
</tr>
<tr>
<td><strong>static vector</strong> <strong>upvec</strong> ()</td>
</tr>
<tr>
<td>return the standard up vector (0, 1, 0)</td>
</tr>
<tr>
<td><strong>static float4</strong> <strong>reciprocal</strong> (const float4 &amp;v)</td>
</tr>
<tr>
<td>return 1.0 / vec</td>
</tr>
<tr>
<td><strong>static float4</strong> <strong>multiply</strong> (const float4 &amp;v0, const float4 &amp;v1)</td>
</tr>
<tr>
<td>component-wise multiplication</td>
</tr>
<tr>
<td><strong>static float4</strong> <strong>cross3</strong> (const float4 &amp;v0, const float4 &amp;v1)</td>
</tr>
<tr>
<td>return 3-dimensional cross product</td>
</tr>
<tr>
<td><strong>static scalar</strong> <strong>dot3</strong> (const float4 &amp;v0, const float4 &amp;v1)</td>
</tr>
<tr>
<td>return 3d dot product of vectors</td>
</tr>
<tr>
<td><strong>static float4</strong> <strong>barycentric</strong> (const float4 &amp;v0, const float4 &amp;v1, const float4 &amp;v2, scalar f, scalar g)</td>
</tr>
<tr>
<td>return point in barycentric coordinates</td>
</tr>
<tr>
<td><strong>static float4</strong> <strong>catmullrom</strong> (const float4 &amp;v0, const float4 &amp;v1, const float4 &amp;v2, const float4 &amp;v3, scalar s)</td>
</tr>
<tr>
<td>perform Catmull-Rom interpolation</td>
</tr>
<tr>
<td><strong>static float4</strong> <strong>hermite</strong> (const float4 &amp;v1, const float4 &amp;t1, const float4 &amp;v2, const float4 &amp;t2, scalar s)</td>
</tr>
<tr>
<td>perform Hermite spline interpolation</td>
</tr>
<tr>
<td><strong>static float4</strong> <strong>lerp</strong> (const float4 &amp;v0, const float4 &amp;v1, scalar s)</td>
</tr>
<tr>
<td>perform linear interpolation between 2 4d vectors</td>
</tr>
<tr>
<td><strong>static float4</strong> <strong>maximize</strong> (const float4 &amp;v0, const float4 &amp;v1)</td>
</tr>
<tr>
<td>return 4d vector made up of largest components of 2 vectors</td>
</tr>
<tr>
<td><strong>static float4</strong> <strong>minimize</strong> (const float4 &amp;v0, const float4 &amp;v1)</td>
</tr>
<tr>
<td>return 4d vector made up of smallest components of 2 vectors</td>
</tr>
<tr>
<td><strong>static float4</strong> <strong>normalize</strong> (const float4 &amp;v)</td>
</tr>
<tr>
<td>return normalized version of 4d vector</td>
</tr>
<tr>
<td><strong>static</strong> <strong>__declspec</strong>(deprecated) float4 <strong>transform</strong>(__Float4Arg v)</td>
</tr>
<tr>
<td>transform 4d vector by matrix44</td>
</tr>
<tr>
<td><strong>static float4</strong> <strong>reflect</strong> (const float4 &amp;normal, const float4 &amp;incident)</td>
</tr>
<tr>
<td>reflect a vector v</td>
</tr>
</tbody>
</table>
static float4 clamp(__Float4Arg Clamp, __Float4Arg vMin, __Float4Arg vMax)
clamp to min/max vector

static scalar angle(__Float4Arg v0, __Float4Arg v1)
angle between two vectors

static bool less3_any (const float4 &v0, const float4 &v1)
return true if any XYZ component is less-then

static bool less3_all (const float4 &v0, const float4 &v1)
return true if all XYZ components are less-then

static bool lessequal3_any (const float4 &v0, const float4 &v1)
return true if any XYZ component is less-or-equal

static bool lessequal3_all (const float4 &v0, const float4 &v1)
return true if all XYZ components are less-or-equal

static bool greater3_any (const float4 &v0, const float4 &v1)
return true if any XYZ component is greater

static bool greater3_all (const float4 &v0, const float4 &v1)
return true if all XYZ components are greater

static bool greaterequal3_any (const float4 &v0, const float4 &v1)
return true if any XYZ component is greater-or-equal

static bool greaterequal3_all (const float4 &v0, const float4 &v1)
return true if all XYZ components are greater-or-equal

static bool equal3_any (const float4 &v0, const float4 &v1)
return true if any XYZ component is equal

static bool equal3_all (const float4 &v0, const float4 &v1)
return true if all XYZ components are equal

static bool nearequal3 (const float4 &v0, const float4 &v1, const float4 &epsilon)
perform near equal comparison with given epsilon (3 components)

static bool less4_any (const float4 &v0, const float4 &v1)
return true if any XYZW component is less-then

static bool less4_all (const float4 &v0, const float4 &v1)
return true if all XYZW components are less-then

static bool lessequal4_any (const float4 &v0, const float4 &v1)
return true if any XYZW component is less-or-equal

static bool lessequal4_all (const float4 &v0, const float4 &v1)
return true if all XYZW components are less-or-equal
greater4\_any (const float4 &v0, const float4 &v1)
return true if any XYZW component is greater

static bool greater4\_all (const float4 &v0, const float4 &v1)
return true if all XYZW components are greater

greaterequal4\_any (const float4 &v0, const float4 &v1)
return true if any XYZW component is greater-or-equal

static bool greaterequal4\_all (const float4 &v0, const float4 &v1)
return true if all XYZW components are greater-or-equal

equal4\_any (const float4 &v0, const float4 &v1)
return true if any XYZW component is equal

equal4\_all (const float4 &v0, const float4 &v1)
return true if all XYZW components are equal

nearequal4 (const float4 &v0, const float4 &v1, const float4 &epsilon)
perform near equal comparison with given epsilon (4 components)

static float unpack\_x (XMVECTOR v)
unpack the first element from a XMVECTOR

static float unpack\_y (XMVECTOR v)
unpack the second element from a XMVECTOR

static float unpack\_z (XMVECTOR v)
unpack the third element from a XMVECTOR

static float unpack\_w (XMVECTOR v)
unpack the fourth element from a XMVECTOR

static float4 splat (scalar s)
splat scalar into each component of a vector

static float4 splat (const float4 &v, uint element)
return a vector with all elements set to element n of v. 0 <= element <= 3

static float4 splat\_x (const float4 &v)
return a vector with all elements set to v.x

static float4 splat\_y (const float4 &v)
return a vector with all elements set to v.y

static float4 splat\_z (const float4 &v)
return a vector with all elements set to v.z

static float4 splat\_w (const float4 &v)
return a vector with all elements set to v.w

static float4 permute\_control (unsigned int i0, unsigned int i1,
unsigned int i2, unsigned int i3)

return control vector for permute (see XMVectorPermuteControl for details)

static float4 permute (const float4 &v0, const float4 &v1, const float4 &control)

merge components of 2 vectors into 1 (see XMVectorPermute for details)
Member Function Documentation

```cpp
__forceinline const void
Math::float4::load (scalar ptr) [inherited]

load content from 16-byte-aligned memory
Load 4 floats from 16-byte-aligned memory.
```

```cpp
__forceinline void
Math::float4::loadu (const scalar ptr) [inherited]

load content from unaligned memory
Load 4 floats from unaligned memory.
```

```cpp
__forceinline void
Math::float4::store (const scalar ptr) const [inherited]

write content to 16-byte-aligned memory through the write cache
Store to 16-byte-aligned float pointer.
```

```cpp
__forceinline void
Math::float4::storeu (scalar ptr) const [inherited]

write content to unaligned memory through the write cache
Store to non-aligned float pointer.
```
Memory::Heap
Memory::Heap Class Reference

#include <heap.h>
Detailed Description

Implements a private heap.

(C) 2006 Radon Labs GmbH
Memory::Memory
#include <memory.h>
Detailed Description

Implements a private heap.

(C) 2006 Radon Labs GmbH
Memory::MemoryPool
Detailed Description

A simple thread-safe memory pool. Memory pool items are 16-byte aligned.

(C) 2009 Radon Labs GmbH
Memory::PoolArrayAllocator
Memory::PoolArrayAllocator Class Reference

#include <poolarrayallocator.h>
Detailed Description

Allocates small memory blocks from an array of fixed-size memory pools. Bigger allocation go directly through to a heap. Note that when freeing a memory block, there are 2 options: one with providing the size of the memory block (which is probably a bit faster) and one conventional without providing the size.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PoolArrayAllocator ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~PoolArrayAllocator ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Setup</strong> (const char <em>name, Memory::HeapType heapType, uint poolSizes[NumPools])</em>*</td>
<td>setup the pool allocator, name must be a static string!</td>
</tr>
<tr>
<td><strong>void * Alloc</strong> (SizeT size)</td>
<td>allocate a block of memory from the pool</td>
</tr>
<tr>
<td><strong>void Free</strong> (void *ptr, SizeT size)</td>
<td>free a block of memory from the pool array with original block size</td>
</tr>
<tr>
<td><strong>void Free</strong> (void *ptr)</td>
<td>free a block of memory from the pool array</td>
</tr>
<tr>
<td>const MemoryPool &amp; <strong>GetMemoryPool</strong> (IndexT index) const</td>
<td>access to memory pool at pool index (for debugging)</td>
</tr>
</tbody>
</table>
### Static Public Attributes

<table>
<thead>
<tr>
<th>static const SizeT</th>
<th>NumPools</th>
<th>= 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>number of pools</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Memory::PoolArrayAllocator::Free ( void * ptr,
    SizeT size
)
```

free a block of memory from the pool array with original block size

This is the faster version to free a memory block, if the caller knows
the size of the memory block we can compute the memory pool index
without asking each pool whether the pointer is owned by this pool.

```cpp
void Memory::PoolArrayAllocator::Free ( void * ptr )
```

free a block of memory from the pool array

This is the slower version to free a memory block. Worst case is, that
the allocator needs to check each memory pool whether the pointer is
owned by the pool.
Memory::TotalMemoryStatus
Memory::TotalMemoryStatus Struct Reference

#include <win360memory.h>
Detailed Description

Get the system's total current memory, this does not only include Nebula3's memory allocations but the memory usage of the entire system.
Messaging::AsyncPort
#include <asyncport.h>

Inheritance diagram for Messaging::AsyncPort:
Detailed Description

The AsyncPort class runs its handlers in a separate thread, so that message processing happens in a separate thread and doesn't block the main thread.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsyncPort()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~AsyncPort()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>SetHandlerThread</code> (const <code>Ptr&lt; HandlerThreadBase &gt;</code> &amp;handlerThread)</td>
<td>set pointer to handler thread object (must be derived from <code>HandlerThreadBase</code>)</td>
</tr>
<tr>
<td><code>GetHandlerThread()</code> const &amp;</td>
<td>get pointer to handler thread object</td>
</tr>
<tr>
<td><code>AttachHandler</code> (const <code>Ptr&lt; Handler &gt;</code> &amp;h)</td>
<td>attach a handler to the port, may be called before or after <code>Open()</code></td>
</tr>
<tr>
<td><code>RemoveHandler</code> (const <code>Ptr&lt; Handler &gt;</code> &amp;h)</td>
<td>dynamically remove a handler from the port</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>open the async port</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>close the async port</td>
</tr>
<tr>
<td><code>IsOpen()</code> const</td>
<td>return true if port is open</td>
</tr>
<tr>
<td><code>Send</code> (const <code>Ptr&lt; MESSAGETYPE &gt;</code> &amp;msg)</td>
<td>send an asynchronous message to the port</td>
</tr>
<tr>
<td><code>SendWait</code> (const <code>Ptr&lt; MESSAGETYPE &gt;</code> &amp;msg)</td>
<td>send a message and wait for completion</td>
</tr>
<tr>
<td><code>Wait</code> (const <code>Ptr&lt; MESSAGETYPE &gt;</code> &amp;msg)</td>
<td>wait for a message to be handled</td>
</tr>
<tr>
<td><code>Peek</code> (const <code>Ptr&lt; MESSAGETYPE &gt;</code> &amp;msg)</td>
<td>peek a message whether it has been handled</td>
</tr>
</tbody>
</table>
template<class MESSAGETYPE>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void Cancel()</td>
<td>cancel a pending message</td>
</tr>
<tr>
<td>int GetRefCount()</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf(const std::string &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA(const std::string &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA(const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const std::string &amp; GetClassName() const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC() const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Messaging::AsyncPort::AttachHandler(const Ptr<Handler> &h) [virtual]
```

attach a handler to the port, may be called before or after `Open()`

Add a message handler, this can either be called before the handler thread is started, or any time afterwards.

Reimplemented in `Interface::InterfaceBase`.

```cpp
void Messaging::AsyncPort::RemoveHandler(const Ptr<Handler> &h) [virtual]
```

dynamically remove a handler from the port

Dynamically remove a message handler.

```cpp
void Messaging::AsyncPort::Open() [virtual]
```

open the async port

Open the async port. The async port needs a valid name before it is opened. Messages can only be sent to an open port.

Reimplemented in `Debug::DebugInterface`, `Http::HttpInterface`, `Interface::InterfaceBase`, and `Graphics::GraphicsInterface`.

```cpp
void Messaging::AsyncPort::Close() [virtual]
```

close the async port

Closes the async port.
Reimplemented in Graphics::GraphicsInterface.

```cpp
int Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Messaging::BatchMessage
#include <batchmessage.h>

Inheritance diagram for Messaging::BatchMessage:
Detailed Description

A batch of messages which is itself a message. Use batch messaging if you want to reduce thread synchronization when sending many messages through an AsyncPort. Instead batch many messages into a single batch message (which doesn't require thread synchronization), then send the batch message as one into the AsyncPort.

Note that the following features don't work on batched messages:

- waiting for the message to become handled
- cancelling the message

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BatchMessage ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>AddMessage (const Ptr&lt; Message&gt; &amp;msg)</strong></td>
<td>Add a message</td>
</tr>
<tr>
<td><strong>GetMessages () const</strong></td>
<td>Read access to message array</td>
</tr>
<tr>
<td><strong>CheckId (const Messaging::Id &amp;id)</strong></td>
<td>Return true if message is of the given id</td>
</tr>
<tr>
<td><strong>Encode (const Ptr&lt; IO::BinaryWriter&gt; &amp;writer)</strong></td>
<td>Encode message into a stream</td>
</tr>
<tr>
<td><strong>Decode (const Ptr&lt; IO::BinaryReader&gt; &amp;reader)</strong></td>
<td>Decode message from a stream</td>
</tr>
<tr>
<td><strong>SetHandled (bool b)</strong></td>
<td>Set the handled flag</td>
</tr>
<tr>
<td><strong>Handled () const</strong></td>
<td>Return true if the message has been handled</td>
</tr>
<tr>
<td><strong>SetDeferred (bool b)</strong></td>
<td>Set deferred flag</td>
</tr>
<tr>
<td><strong>IsDeferred () const</strong></td>
<td>Get deferred flag</td>
</tr>
<tr>
<td><strong>SetDeferredHandled (bool b)</strong></td>
<td>Set the deferred handled flag</td>
</tr>
<tr>
<td><strong>DeferredHandled () const</strong></td>
<td>Get the deferred handled flag</td>
</tr>
<tr>
<td><strong>GetRefCount () const</strong></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td></td>
</tr>
</tbody>
</table>
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Messaging::BlockingHandlerThread
#include <blockinghandlerthread.h>

Inheritance diagram for Messaging::BlockingHandlerThread:

```
Core::RefCounted
  |
  v
Win32::Win32Thread
  |
  v
Thread
  |
  v
Messaging::HandlerThreadBase
  |
  v
Messaging::BlockingHandlerThread
```
Detailed Description

Message handler thread class which blocks until messages arrive (or optionally, a time-out occurs).

(C) 2009 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>thread priorities</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BlockingHandlerThread()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>void SetWaitTimeout(int milliSec)</code></td>
<td>set optional wait timeout (0 if infinite)</td>
</tr>
<tr>
<td><code>int GetWaitTimeout()</code> const</td>
<td>get wait timeout</td>
</tr>
<tr>
<td><code>virtual void AddMessage(const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>add a message to be handled (override in subclass!)</td>
</tr>
<tr>
<td><code>virtual void CancelMessage(const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>cancel a pending message (override in subclass!)</td>
</tr>
<tr>
<td><code>virtual void EmitWakeupSignal()</code></td>
<td>called if thread needs a wakeup call before stopping</td>
</tr>
<tr>
<td><code>virtual void DoWork()</code></td>
<td>this method runs in the thread context</td>
</tr>
<tr>
<td><code>void AttachHandler(const Ptr&lt;Handler&gt; &amp;h)</code></td>
<td>attach a message handler</td>
</tr>
<tr>
<td><code>void RemoveHandler(const Ptr&lt;Handler&gt; &amp;h)</code></td>
<td>dynamically remove a message handler</td>
</tr>
<tr>
<td><code>void ClearHandlers()</code></td>
<td>clear all attached message handlers</td>
</tr>
<tr>
<td><code>void WaitForHandlersOpened()</code></td>
<td>wait until handlers have been opened</td>
</tr>
<tr>
<td><code>virtual void WaitForMessage(const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>wait for message to be handled (optionally override in subclass!)</td>
</tr>
<tr>
<td><code>void SetPriority(Priority p)</code></td>
<td>set the thread priority</td>
</tr>
<tr>
<td><code>Priority GetPriority()</code> const</td>
<td>get the thread priority</td>
</tr>
<tr>
<td><code>void SetCoreId(System::Cpu::CoreId coreId)</code></td>
<td>set cpu core on which the thread should be running</td>
</tr>
<tr>
<td><code>System::Cpu::CoreId GetCoreId()</code> const</td>
<td>get the cpu core on which the thread should be running</td>
</tr>
<tr>
<td><code>void SetStackSize(SizeT s)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>SizeT GetStackSize () const</code></td>
<td>get stack size</td>
</tr>
<tr>
<td><code>void SetName (const Util::String &amp;n)</code></td>
<td>set thread name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetName () const</code></td>
<td>get thread name</td>
</tr>
<tr>
<td><code>void Start ()</code></td>
<td>start executing the thread code, returns when thread has actually started</td>
</tr>
<tr>
<td><code>void Stop ()</code></td>
<td>request threading code to stop, returns when thread has actually finished</td>
</tr>
<tr>
<td><code>bool IsRunning () const</code></td>
<td>return true if thread has been started</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get class name</td>
</tr>
<tr>
<td>get the class name</td>
<td>Util::FourCC</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void YieldThread()</code></td>
<td>yield the thread (gives up current time slice)</td>
</tr>
<tr>
<td><code>static void SetMyThreadName(const char *n)</code></td>
<td>set thread name from within thread context</td>
</tr>
<tr>
<td><code>static const char * GetMyThreadName()</code></td>
<td>obtain name of thread from within thread context</td>
</tr>
<tr>
<td><code>static Threading::ThreadId GetMyThreadId()</code></td>
<td>get the thread ID of this thread</td>
</tr>
<tr>
<td><code>static void DumpRefCountingLeaks()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void ThreadOpenHandlers ()</code></td>
<td>open message handlers</td>
</tr>
<tr>
<td><code>void ThreadCloseHandlers ()</code></td>
<td>close message handlers</td>
</tr>
<tr>
<td><code>void ThreadUpdateHandlers ()</code></td>
<td>open dynamically added handlers, and call DoWork() on all attached handlers</td>
</tr>
<tr>
<td><code>bool ThreadUpdateDeferredMessages ()</code></td>
<td>update deferred messages, return true if at least one message has been handled</td>
</tr>
<tr>
<td><code>void ThreadDiscardDeferredMessages ()</code></td>
<td>clear leftover deferred messages</td>
</tr>
<tr>
<td><code>bool ThreadHandleMessages (const Util::Array&lt; Ptr&lt; Message &gt; &gt; &amp;msgArray)</code></td>
<td>handle messages in array, return true if at least one message has been handled</td>
</tr>
<tr>
<td><code>bool ThreadHandleSingleMessage (const Ptr&lt; Message &gt; &amp;msg)</code></td>
<td>handle a single message without deferred support, return if message has been handled</td>
</tr>
<tr>
<td><code>void ThreadSignalMessageHandled ()</code></td>
<td>signal message handled event (only call if at least one message has been handled)</td>
</tr>
<tr>
<td><code>bool ThreadStopRequested () const</code></td>
<td>check if stop is requested, call from DoWork() to see if the thread proc should quit</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Messaging::BlockingHandlerThread::AddMessage(const Ptr<Message>& msg) [virtual]
```
add a message to be handled (override in subclass!)
This adds a new message to the thread's message queue.
Reimplemented from `Messaging::HandlerThreadBase`.

```cpp
void Messaging::BlockingHandlerThread::CancelMessage(const Ptr<Message>& msg) [virtual]
cancel a pending message (override in subclass!)
This removes a message from the thread's message queue, regardless of its state.
Reimplemented from `Messaging::HandlerThreadBase`.
```

```cpp
void Messaging::BlockingHandlerThread::EmitWakeupSignal( ) [virtual]
called if thread needs a wakeup call before stopping
This method is called by `Thread::Stop()` after setting the stopRequest event and before waiting for the thread to stop.
Reimplemented from `Win360::Win360Thread`.
```

```cpp
void Messaging::BlockingHandlerThread::DoWork( ) [virtual]
this method runs in the thread context
The message processing loop.
Reimplemented from **Win360::Win360Thread**.

```cpp
void Messaging::HandlerThreadBase::AttachHandler(
    const Ptr<Handler>& h ) [inherited]
```

attach a message handler

Attach a message handler to the port. This method may be called from any thread.

```cpp
void Messaging::HandlerThreadBase::RemoveHandler(
    const Ptr<Handler>& h ) [inherited]
```

dynamically remove a message handler

Remove a message handler. This method may be called form any thread.

```cpp
void Messaging::HandlerThreadBase::ClearHandlers( ) [inherited]
```

clear all attached message handlers

This clears all attached message handlers.

```cpp
void Messaging::HandlerThreadBase::WaitForHandlersOpened( ) [inherited]
```

wait until handlers have been opened

Wait on the handlers-opened event (will be signalled by the ThreadOpenHandlers method.

```cpp
void Messaging::HandlerThreadBase::WaitForMessage(
    const Ptr<Message>& msg ) [virtual, inherited]
```

wait for message to be handled (optionally override in subclass!)
This waits until the message given as argument has been handled. In order for this message to work, the `ThreadSignalMessageHandled()` must be called from within the handler thread context. Note that subclasses may override this method if they need to.

Reimplemented in `FrameSync::FrameSyncHandlerThread`.

```cpp
void Messaging::HandlerThreadBase::ThreadOpenHandlers( ) [protected, inherited]
```

`open message handlers`

Open attached message handlers. This method must be called at the start of the handler thread.

```cpp
void Messaging::HandlerThreadBase::ThreadCloseHandlers( ) [protected, inherited]
```

`close message handlers`

Close attached message handlers. This method must be called right before the handler thread shuts down.

```cpp
void Messaging::HandlerThreadBase::ThreadUpdateHandlers( ) [protected, inherited]
```

`open dynamically added handlers, and call `DoWork()` on all attached handlers`

Do per-frame update of attached handlers. This will open handlers which have been added late, and call the `DoWork()` method on handlers from within the thread context.

```cpp
bool Messaging::HandlerThreadBase::ThreadUpdateDeferredMessages( ) [protected, inherited]
```

`update deferred messages, return true if at least one message has been handled`

This checks every message in the deferred message array whether it has been handled yet, if yes, the message's actual handled flag will be
set, and the message will be removed from the deferred handled array. If at least one message has been handled, the method will return true, if no message has been handled, the method returns false. If message have been handled, don’t forget to call `ThreadSignalMessageHandled()` later!

```cpp
void Messaging::HandlerThreadBase::ThreadDiscardDeferredMessages() [protected, inherited]
```

clear leftover deferred messages

This clears any leftover deferred messages. Call right before shutdown of the handler thread.

```cpp
bool Messaging::HandlerThreadBase::ThreadHandleMessages(const Util::Array<Ptr<Message>> &msgArray) [protected, inherited]
```

handle messages in array, return true if at least one message has been handled

Handle all message in the provided message array. Supports batched and deferred messages. Calls `ThreadHandleSingleMessage()`. If at least one message has been handled, the method returns true.

```cpp
bool Messaging::HandlerThreadBase::ThreadHandleSingleMessage(const Ptr<Message> &msg) [protected, inherited]
```

handle a single message without deferred support, return if message has been handled

Handle a single message, called by `ThreadHandleMessages()`. Return true if message has been handled. This method MUST be called from `ThreadHandleMessages()`, since this method will not explicitely take the handlers array critical section.

```cpp
void Messaging::HandlerThreadBase::ThreadSignalMessageHandled() [protected, inherited]
```
signal message handled event (only call if at least one message has been handled)

Signal the message-handled flag. Call this method once per handler-loop if either ThreadUpdateDeferredMessages or ThreadHandleMessages returns true!

```cpp
void Win360::Win360Thread::SetName(const Util::String & n ) [inline, inherited]
```

set thread name

Set the thread's name. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

```cpp
const Util::String & Win360::Win360Thread::GetName( ) const [inline, inherited]
```

get thread name

Get the thread's name. This is the vanilla method which returns the name member. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

```cpp
void Win360::Win360Thread::Start( ) [inherited]
```

start executing the thread code, returns when thread has actually started

Start the thread, this creates a Win32 thread and calls the static ThreadProc, which in turn calls the virtual `DoWork()` class of this object. The method waits for the thread to start and then returns.

```cpp
void Win360::Win360Thread::Stop( ) [inherited]
```

request threading code to stop, returns when thread has actually finished
This stops the thread by signalling the stopRequestEvent and waits for the thread to actually quit. If the thread code runs in a loop it should use the IsStopRequested() method to see if the thread object wants it to shutdown. If so `DoWork()` should simply return.

Reimplemented in `Jobs::TPWorkerThread`.

```cpp
bool Win360::Win360Thread::IsRunning() const [inherited]
```

return true if thread has been started

Returns true if the thread is currently running.

```cpp
void Win360::Win360Thread::YieldThread() [static, inherited]
```

yield the thread (gives up current time slice)

The yield function is empty on `Win32` and `Xbox360`.

```cpp
void Win360::Win360Thread::SetMyThreadName(const char * n ) [static, inherited]
```

set thread name from within thread context

Static method which sets the name of this thread. This is called from within ThreadProc. The string pointed to must remain valid until the thread is terminated!

```cpp
const char * Win360::Win360Thread::GetMyThreadName() [static, inherited]
```

obtain name of thread from within thread context

Static method to obtain the current thread name from anywhere in the thread's code.

```cpp
Threading::ThreadId Win360::Win360Thread::GetMyThreadId() [static, inherited]
```
get the thread ID of this thread

Static method which returns the ThreadId of this thread.

```cpp
bool Win360::Win360Thread::ThreadStopRequested() const [inline, protected, inherited]
```

check if stop is requested, call from `DoWork()` to see if the thread proc should quit

If the derived `DoWork()` method is running in a loop it must regularly check if the process wants the thread to terminate by calling `ThreadStopRequested()` and simply return if the result is true. This will cause the thread to shut down.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.
Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Messaging::<b>DelegateTable</b>
Messaging::DelegateTable Class Reference

#include <delegatetable.h>
Detailed Description

Associates message ids with handler delegates. One message id may be associated with any number of handler functions. The order of handler functions for one message id will be preserved.

(C) 2008 Radon Labs GmbH
Public Member Functions

template<class CLASS, void(CLASS::*)(const Ptr<Message> &)> METHOD>
void Bind (const Id &msgId, CLASS *obj)  
bind a message id to a method call

bool Invoke (const Ptr<Message> &msg)  
invoke delegates bound to msg, returns false if no delegates bound to msg id
Messaging::Dispatcher
#include <dispatcher.h>

Inheritance diagram for Messaging::Dispatcher:
Detailed Description

A message **Dispatcher** is a specialization of a message **Port**. A message **Dispatcher** distributes all messages it receives to the attached Ports which are interested in this message id.

**Dispatcher** objects usually serve as front end message ports which hide a more complex message processing infrastructure underneath.

(C) 2007 RadonLabs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatcher ( )</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual void HandleMessage (const Pointer<a href="">Messaging::Message</a> &amp;msg)</td>
<td>handle a single message (distribute to ports which accept the message)</td>
</tr>
<tr>
<td>void AttachPort (const Pointer&lt;Port&gt; &amp;port)</td>
<td>attach a message port</td>
</tr>
<tr>
<td>void RemovePort (const Pointer&lt;Port&gt; &amp;port)</td>
<td>remove a message port</td>
</tr>
<tr>
<td>bool HasPort (const Pointer&lt;Port&gt; &amp;port) const</td>
<td>return true if a port exists</td>
</tr>
<tr>
<td>virtual void SetupAcceptedMessages ( )</td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>void AttachHandler (const Pointer&lt;Handler&gt; &amp;h)</td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td>void RemoveHandler (const Pointer&lt;Handler&gt; &amp;h)</td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td>void RemoveAllHandlers ( )</td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td>SizeT GetNumHandlers ( ) const</td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td>const Pointer&lt;Handler&gt; &amp; GetHandlerAtIndex (IndexT i) const</td>
<td>get a message handler by index</td>
</tr>
<tr>
<td>virtual void Send (const Pointer&lt;Message&gt; &amp;msg)</td>
<td>send a message to the port</td>
</tr>
<tr>
<td>const Util::Array&lt; const Id * &gt; &amp; GetAcceptedMessages ( ) const</td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td>bool AcceptsMessage (const Id &amp;msgId) const</td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td>int GetRefCount ( ) const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName() const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC() const</code></td>
<td>Get the class FourCC code</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBU {L3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
void RegisterMessage (const Id &msgId)
  register a single accepted message
```
Member Function Documentation

```cpp
void Messaging::Dispatcher::HandleMessage(const Messaging::Message msg) [virtual]
```

handle a single message (distribute to ports which accept the message)

Handle a message. The message will only be distributed to ports which accept the message.

Reimplemented from `Messaging::Port`.

Reimplemented in `Script::DialogManager`.

```cpp
void Messaging::Dispatcher::AttachPort(const Port& port)
```

attach a message port

Attach a new message port.

**Parameters:**

- `port` pointer to a message port object

```cpp
void Messaging::Dispatcher::RemovePort(const Port& port)
```

remove a message port

Remove a message port object.

**Parameters:**

- `handler` pointer to message port object to be removed

  ```cpp
  const
  ```
bool Messaging::Dispatcher::HasPort (Ptr<Port> & port) const

return true if a port exists

Return true if a port is already attached.

void Messaging::Port::AttachHandler (const Ptr<Handler> & h) [inherited]

attach a message handler to the port

Attach a message handler to the port.

void Messaging::Port::RemoveHandler (const Ptr<Handler> & h) [inherited]

remove a message handler from the port

Remove a message handler from the port.

void Messaging::Port::Send (const Ptr<Message> & msg) [virtual, inherited]

send a message to the port

Send a message to the port. This will immediately call the HandleMessage() method of all attached handlers. If the message has been handled by at least one of the handlers, the Handled() flag of the message will be set to true.

int Core::RefCounted::GetRefCount () const [inline, inherited]

get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Messaging::Handler
#include <handler.h>

Inheritance diagram for Messaging::Handler:
Detailed Description

**Message** handlers are used to process a message. To handle specific messages, derive from **Handler** and overwrite the method **HandleMessage()**.

(C) 2006 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Handler ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~Handler ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <strong>Open ()</strong></td>
<td>called once on startup</td>
</tr>
<tr>
<td>virtual void <strong>Close ()</strong></td>
<td>called once before shutdown</td>
</tr>
<tr>
<td>bool <strong>IsOpen () const</strong></td>
<td>return true if open</td>
</tr>
<tr>
<td>virtual bool <strong>HandleMessage</strong> (const <strong>Ptr&lt; Message &gt;</strong> &amp;msg)</td>
<td>handle a message, return true if handled</td>
</tr>
<tr>
<td>virtual void <strong>DoWork ()</strong></td>
<td>optional &quot;per-frame&quot; DoWork method for continuous handlers</td>
</tr>
<tr>
<td>int <strong>GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOF</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOF</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOF</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><code>GetClassName () const</code></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><code>GetClassFourCC () const</code></td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

void Messaging::Handler::Open ( ) [virtual]
called once on startup

Open the handler. This method is called once after the handler has been attached to a port and before the first call to HandleMessage().

Reimplemented in Debug::DebugHandler, Http::HttpMessageHandler, IO::IoInterfaceHandler, Animation::AnimEventServer, and Graphics::GraphicsHandler.

void Messaging::Handler::Close ( ) [virtual]
called once before shutdown

Close the handler. This method is called once before the handler is detached from the port.

Reimplemented in Debug::DebugHandler, Http::HttpMessageHandler, IO::IoInterfaceHandler, Animation::AnimEventServer, and Graphics::GraphicsHandler.

bool Messaging::Handler::HandleMessage ( const Ptr< Message > & msg ) [virtual]

handle a message, return true if handled

Derive this method in a subclass to handle specific messages. The method should return true only if a message has been handled.

Reimplemented in Http::HttpMessageHandler, IO::IoInterfaceHandler, Animation::AnimEventServer, Debug::DebugGraphicsHandler, and Graphics::GraphicsHandler.
void Messaging::Handler::DoWork();

optional "per-frame" DoWork method for continuous handlers

This is an optional method for handlers which need to do continuous work (like a render thread message handler). This message will be called after messages have been handled.

Reimplemented in Debug::DebugHandler, Http::HttpMessageHandler, Interface::InterfaceHandlerBase, Debug::DebugGraphicsHandler, and Graphics::GraphicsHandler.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC() const [inline, inherited]
Core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
Messaging::HandlerThreadBase
#include <handlerthreadbase.h>

Inheritance diagram for Messaging::HandlerThreadBase:

```
Core::RefCounted
  |
  v
Win32::Win32Thread
  |
  v
Threading::Thread
  |
  v
Messaging::HandlerThreadBase
  |
FrameSync::FrameSyncHandlerThread  Messaging::BlockingHandlerThread  Messaging::RunThroughHandlerThread
```
Detailed Description

**Base** class for **AsyncPort** message handler thread classes.

(C) 2009 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>thread priorities</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HandlerThreadBase</strong>()</td>
<td>constructor</td>
</tr>
<tr>
<td><strong>void AttachHandler (const Ptr&lt; Handler&gt; &amp;h)</strong></td>
<td>attach a message handler</td>
</tr>
<tr>
<td><strong>void RemoveHandler (const Ptr&lt; Handler&gt; &amp;h)</strong></td>
<td>dynamically remove a message handler</td>
</tr>
<tr>
<td><strong>void ClearHandlers ()</strong></td>
<td>clear all attached message handlers</td>
</tr>
<tr>
<td><strong>void WaitForHandlersOpened ()</strong></td>
<td>wait until handlers have been opened</td>
</tr>
<tr>
<td><strong>virtual void AddMessage (const Ptr&lt; Message&gt; &amp;msg)</strong></td>
<td>add a message to be handled (override in subclass!)</td>
</tr>
<tr>
<td><strong>virtual void CancelMessage (const Ptr&lt; Message&gt; &amp;msg)</strong></td>
<td>cancel a pending message (override in subclass!)</td>
</tr>
<tr>
<td><strong>virtual void WaitForMessage (const Ptr&lt; Message&gt; &amp;msg)</strong></td>
<td>wait for message to be handled (optionally override in subclass!)</td>
</tr>
<tr>
<td><strong>void SetPriority (Priority p)</strong></td>
<td>set the thread priority</td>
</tr>
<tr>
<td><strong>Priority GetPriority () const</strong></td>
<td>get the thread priority</td>
</tr>
<tr>
<td><strong>void SetCoreId (System::Cpu::CoreId coreId)</strong></td>
<td>set cpu core on which the thread should be running</td>
</tr>
<tr>
<td><strong>System::Cpu::CoreId GetCoreId () const</strong></td>
<td>get the cpu core on which the thread should be running</td>
</tr>
<tr>
<td><strong>void SetStackSize (SizeT s)</strong></td>
<td>set stack size in bytes (default is 4 KByte)</td>
</tr>
<tr>
<td><strong>SizeT GetStackSize () const</strong></td>
<td>get stack size</td>
</tr>
<tr>
<td><strong>void SetName (const Util::String &amp;n)</strong></td>
<td>set thread name</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetName () const</strong></td>
<td>get thread name</td>
</tr>
<tr>
<td><strong>void Start ()</strong></td>
<td></td>
</tr>
</tbody>
</table>
start executing the thread code, returns when thread has actually started

```cpp
void Stop ()
request threading code to stop, returns when thread has actually finished
```

```cpp
bool IsRunning () const
return true if thread has been started
```

```cpp
int GetRefCount () const
get the current refcount
```

```cpp
void AddRef ()
increment refcount by one
```

```cpp
void Release ()
decrement refcount and destroy object if refcount is zero
```

```cpp
bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class
```

```cpp
bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string
```

```cpp
bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc
```

```cpp
bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class
```

```cpp
bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string
```

```cpp
bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc
```

```cpp
const Util::String & GetClassName () const
get the class name
```

```cpp
Util::FourCC GetClassFourCC () const
get the class FourCC code
```
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void YieldThread ()</code></td>
<td>yield the thread (gives up current time slice)</td>
</tr>
<tr>
<td><code>static void SetMyThreadName (const char *n)</code></td>
<td>set thread name from within thread context</td>
</tr>
<tr>
<td><code>static const char * GetMyThreadName ()</code></td>
<td>obtain name of thread from within thread context</td>
</tr>
<tr>
<td><code>static Threading::ThreadId GetMyThreadId ()</code></td>
<td>get the thread ID of this thread</td>
</tr>
<tr>
<td><code>static void DumpRefCountingLeaks ()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void ThreadOpenHandlers()</code></td>
<td>Open message handlers</td>
</tr>
<tr>
<td><code>void ThreadCloseHandlers()</code></td>
<td>Close message handlers</td>
</tr>
<tr>
<td><code>void ThreadUpdateHandlers()</code></td>
<td>Open dynamically added handlers, and call <code>DoWork()</code> on all attached handlers</td>
</tr>
<tr>
<td><code>bool ThreadUpdateDeferredMessages()</code></td>
<td>Update deferred messages, return <code>true</code> if at least one message has been handled</td>
</tr>
<tr>
<td><code>void ThreadDiscardDeferredMessages()</code></td>
<td>Clear leftover deferred messages</td>
</tr>
<tr>
<td><code>bool ThreadHandleMessages(const Util::Array&lt; Ptr&lt;Message&gt; &gt; &amp;msgArray)</code></td>
<td>Handle messages in array, return <code>true</code> if at least one message has been handled</td>
</tr>
<tr>
<td><code>bool ThreadHandleSingleMessage(const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>Handle a single message without deferred support, return <code>true</code> if message has been handled</td>
</tr>
<tr>
<td><code>void ThreadSignalMessageHandled()</code></td>
<td>Signal message handled event (only call if at least one message has been handled)</td>
</tr>
<tr>
<td><code>virtual void EmitWakeupSignal()</code></td>
<td>Override this method if your thread loop needs a wakeup call before stopping</td>
</tr>
<tr>
<td><code>virtual void DoWork()</code></td>
<td>This method runs in the thread context</td>
</tr>
<tr>
<td><code>bool ThreadStopRequested()</code> const</td>
<td>Check if stop is requested, call from <code>DoWork()</code> to see if the thread proc should quit</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Messaging::HandlerThreadBase::AttachHandler(const Ptr<Handler> &h)

attach a message handler

Attach a message handler to the port. This method may be called from any thread.

void Messaging::HandlerThreadBase::RemoveHandler(const Ptr<Handler> &h)

dynamically remove a message handler

Remove a message handler. This method may be called from any thread.

void Messaging::HandlerThreadBase::ClearHandlers()

clear all attached message handlers

This clears all attached message handlers.

void Messaging::HandlerThreadBase::WaitForHandlersOpened()

wait until handlers have been opened

Wait on the handlers-opened event (will be signalled by the ThreadOpenHandlers method.)

void Messaging::HandlerThreadBase::AddMessage(const Ptr<Message> &msg)[virtual]

add a message to be handled (override in subclass!)

This adds a new message to the thread's message queue. In the base class this message is empty and must be implemented in a subclass.

Reimplemented in FrameSync::FrameSyncHandlerThread, Messaging::BlockingHandlerThread, and Messaging::RunThroughHandlerThread.

```cpp
void Messaging::HandlerThreadBase::CancelMessage(const Ptr<Message> &msg) [virtual]
```

cancel a pending message (override in subclass!)

This removes a message from the thread's message queue, regardless of its state. Override this method in a subclass.

Reimplemented in FrameSync::FrameSyncHandlerThread, Messaging::BlockingHandlerThread, and Messaging::RunThroughHandlerThread.

```cpp
void Messaging::HandlerThreadBase::WaitForMessage(const Ptr<Message> &msg) [virtual]
```

wait for message to be handled (optionally override in subclass!)

This waits until the message given as argument has been handled. In order for this message to work, the ThreadSignalMessageHandled() must be called from within the handler thread context. Note that subclasses may override this method if they need to.

Reimplemented in FrameSync::FrameSyncHandlerThread.

```cpp
void Messaging::HandlerThreadBase::ThreadOpenHandlers( ) [protected]
```

open message handlers
Open attached message handlers. This method must be called at the start of the handler thread.

```c++
void Messaging::HandlerThreadBase::ThreadCloseHandlers()
```

**close message handlers**

Close attached message handlers. This method must be called right before the handler thread shuts down.

```c++
void Messaging::HandlerThreadBase::ThreadUpdateHandlers()
```

**open dynamically added handlers, and call DoWork() on all attached handlers**

Do per-frame update of attached handlers. This will open handlers which have been added late, and call the `DoWork()` method on handlers from within the thread context.

```c++
bool Messaging::HandlerThreadBase::ThreadUpdateDeferredMessages()
```

**update deferred messages, return true if at least one message has been handled**

This checks every message in the deferred message array whether it has been handled yet, if yes, the message's actual handled flag will be set, and the message will be removed from the deferred handled array. If at least one message has been handled, the method will return true, if no message has been handled, the method returns false. If message have been handled, don't forget to call `ThreadSignalMessageHandled()` later!

```c++
void Messaging::HandlerThreadBase::ThreadDiscardDeferredMessages()
```

**clear leftover deferred messages**

This clears any leftover deferred messages. Call right before
shutdown of the handler thread.

```cpp
bool Messaging::HandlerThreadBase::ThreadHandleMessages(
    const Util::Array<Ptr<Message>>& msgArray) [protected]
```

handle messages in array, return true if at least one message has been handled

Handle all message in the provided message array. Supports batched and deferred messages. Calls `ThreadHandleSingleMessage()`. If at least one message has been handled, the method returns true.

```cpp
bool Messaging::HandlerThreadBase::ThreadHandleSingleMessage(
    const Ptr<Message>& msg) [protected]
```

handle a single message without deferred support, return if message has been handled

Handle a single message, called by `ThreadHandleMessages()`. Return true if message has been handled. This method MUST be called from `ThreadHandleMessages()`, since this method will not explicitly take the handlers array critical section.

```cpp
void Messaging::HandlerThreadBase::ThreadSignalMessageHandled() [protected]
```

signal message handled event (only call if at least one message has been handled)

Signal the message-handled flag. Call this method once per handler-loop if either `ThreadUpdateDeferredMessages` or `ThreadHandleMessages` returns true!

```cpp
void Win360::Win360Thread::SetName(
    const Util::String n) [inline, inherited]
```

set thread name
Set the thread's name. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

```cpp
const Util::String & Win360::Win360Thread::GetName() const [inline, inherited]
```

get thread name

Get the thread's name. This is the vanilla method which returns the name member. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

```cpp
void Win360::Win360Thread::Start() [inherited]
```

start executing the thread code, returns when thread has actually started

Start the thread, this creates a Win32 thread and calls the static ThreadProc, which in turn calls the virtual `DoWork()` class of this object. The method waits for the thread to start and then returns.

```cpp
void Win360::Win360Thread::Stop() [inherited]
```

request threading code to stop, returns when thread has actually finished

This stops the thread by signalling the stopRequestEvent and waits for the thread to actually quit. If the thread code runs in a loop it should use the IsStopRequested() method to see if the thread object wants it to shutdown. If so `DoWork()` should simply return.

Reimplemented in `Jobs::TPWorkerThread`.

```cpp
bool Win360::Win360Thread::IsRunning() const [inherited]
```

return true if thread has been started
Returns true if the thread is currently running.

```cpp
void
Win360::Win360Thread::YieldThread() [static, inherited]
```

yield the thread (gives up current time slice)

The yield function is empty on **Win32** and Xbox360.

```cpp
void
Win360::Win360Thread::SetMyThreadName(const char * n) [static, inherited]
```

set thread name from within thread context

Static method which sets the name of this thread. This is called from within ThreadProc. The string pointed to must remain valid until the thread is terminated!

```cpp
const char *
Win360::Win360Thread::GetMyThreadName() [static, inherited]
```

obtain name of thread from within thread context

Static method to obtain the current thread name from anywhere in the thread's code.

```cpp
ThreadId
Win360::Win360Thread::GetMyThreadId() [static, inherited]
```

get the thread ID of this thread

Static method which returns the ThreadId of this thread.

```cpp
void
Win360::Win360Thread::EmitWakeupSignal() [protected, virtual, inherited]
```

override this method if your thread loop needs a wakeup call before stopping

This method is called by **Thread::Stop()** after setting the stopRequest event and before waiting for the thread to stop. If your thread runs a
loop and waits for jobs it may need an extra wakeup signal to stop waiting and check for the `ThreadStopRequested()` event. In this case, override this method and signal your event object.

Reimplemented in `Jobs::TPWorkerThread`, and `Messaging::BlockingHandlerThread`.

```cpp
void Win360::Win360Thread::DoWork() [protected, virtual, inherited]
```

This method should be derived in a Thread subclass and contains the actual code which is run in the thread. The method must not call C-Lib functions under `Win32`. To terminate the thread, just return from this function. If `DoWork()` runs in an infinite loop, call `ThreadStopRequested()` to check whether the Thread object wants the thread code to quit.

Reimplemented in `FrameSync::FrameSyncHandlerThread`, `Jobs::TPWorkerThread`, `Messaging::BlockingHandlerThread`, and `Messaging::RunThroughHandlerThread`.

```cpp
bool Win360::Win360Thread::ThreadStopRequested() const [inline, protected, inherited]
```

check if stop is requested, call from `DoWork()` to see if the thread proc should quit

If the derived `DoWork()` method is running in a loop it must regularly check if the process wants the thread to terminate by calling `ThreadStopRequested()` and simply return if the result is true. This will cause the thread to shut down.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Messaging::Id
Messaging::Id Class Reference

#include <id.h>
Detailed Description

A message identifier. This is automatically implemented in message classes using the __DeclareMsgId and __ImplementMsgId macros.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Id ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>constructor</em></td>
</tr>
</tbody>
</table>

| bool operator== (const Id &rhs) const |
| *equality operator* |
Messaging::Message
#include <message.h>

Inheritance diagram for Messaging::Message:
Detailed Description

Messages are packets of data which can be sent to a message port. This implements a universal communication mechanism within the same thread, different threads, or even different machines.

Messages are implemented as normal C++ objects which can encode and decode themselves from and to a stream.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Message ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>bool CheckId (const Messaging::Id &amp;id) const</strong></td>
<td>return true if message is of the given id</td>
</tr>
<tr>
<td><strong>virtual void Encode (const Ptr&lt; IO::BinaryWriter &gt; &amp;writer)</strong></td>
<td>encode message into a stream</td>
</tr>
<tr>
<td><strong>virtual void Decode (const Ptr&lt; IO::BinaryReader &gt; &amp;reader)</strong></td>
<td>decode message from a stream</td>
</tr>
<tr>
<td><strong>void SetHandled (bool b)</strong></td>
<td>set the handled flag</td>
</tr>
<tr>
<td><strong>bool Handled () const</strong></td>
<td>return true if the message has been handled</td>
</tr>
<tr>
<td><strong>void SetDeferred (bool b)</strong></td>
<td>set deferred flag</td>
</tr>
<tr>
<td><strong>bool IsDeferred () const</strong></td>
<td>get deferred flag</td>
</tr>
<tr>
<td><strong>void SetDeferredHandled (bool b)</strong></td>
<td>set the deferred handled flag</td>
</tr>
<tr>
<td><strong>bool DeferredHandled () const</strong></td>
<td>get the deferred handled flag</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className)</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>return true if this object is instance of given class by FourCC</td>
</tr>
</tbody>
</table>
return true if this object is instance of given class by fourcc

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```c++
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```c++
void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```c++
void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```c++
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

```c++
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```c++
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Messaging::MessageReader
#include <messagereader.h>

Inheritance diagram for Messaging::MessageReader:
Detailed Description

Implements a binary stream protocol for decoding messages from a stream.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MessageReader ()</strong>&lt;br&gt;constructor</td>
<td></td>
</tr>
<tr>
<td>virtual void <strong>SetStream</strong> (const <strong>Ptr&lt; IO::Stream &gt;</strong> &amp;s)</td>
<td>set stream to read from</td>
</tr>
<tr>
<td>**Message * ** <strong>ReadMessage</strong> ()</td>
<td>decode a new message from the stream</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; Stream &gt;</strong> &amp; <strong>GetStream</strong> () const</td>
<td>get currently set stream</td>
</tr>
<tr>
<td>bool <strong>HasStream</strong> () const</td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td>bool <strong>Eof</strong> () const</td>
<td>return true if the stream has reached EOF</td>
</tr>
<tr>
<td>virtual bool <strong>Open</strong> ()</td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td>virtual void <strong>Close</strong> ()</td>
<td>end reading from the stream</td>
</tr>
<tr>
<td>bool <strong>IsOpen</strong> () const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className)</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td><em>dump refcounting leaks, call at end of application</em> (NEBULA3_DEBUG builds only!)</td>
<td></td>
</tr>
</tbody>
</table>
Message Documentation

**Message** * 
Messaging::MessageReader::ReadMessage()

decode a new message from the stream

This constructs a new message from the stream. First the FourCC class id of the message will be read, and a new message object constructed from it, then the message object will be asked to initialize itself from the stream.

```cpp
const Ptr< Stream > &
IO::StreamReader::GetStream() const [inherited]
```

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use HasStream() to determine if a stream is attached.

```cpp
bool
IO::StreamReader::HasStream() const [inherited]
```

return true if a stream is set

Returns true if a stream is attached to the reader.

```cpp
int
Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void
Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String& Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Messaging::MessageWriter
#include <messagewriter.h>

Inheritance diagram for Messaging::MessageWriter:

```
Core::RefCounted

IO::StreamWriter

Messaging::MessageWriter
```
Detailed Description

Implements a binary stream protocol for encoding messages into streams.

(C) 2006 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MessageWriter ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual void SetStream (const Ptr&lt; IO::Stream &gt; &amp;s)</strong></td>
<td>set stream to write to</td>
</tr>
<tr>
<td><strong>void WriteMessage (const Ptr&lt; Message &gt; &amp;msg)</strong></td>
<td>write a complete message to the stream</td>
</tr>
<tr>
<td><strong>const Ptr&lt; Stream &gt; &amp; GetStream () const</strong></td>
<td>get currently set stream</td>
</tr>
<tr>
<td><strong>bool HasStream () const</strong></td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td><strong>virtual bool Open ()</strong></td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td><strong>virtual void Close ()</strong></td>
<td>end reading from the stream</td>
</tr>
<tr>
<td><strong>bool IsOpen () const</strong></td>
<td>return true if currently open</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className)</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>bool IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Messaging::MessageWriter::WriteMessage (const Ptr<Message>& msg)
```

write a complete message to the stream

Writes a complete message to the stream. First the FourCC class id of the message will be written, then the message will be asked to write its own data to the stream.

```cpp
const Ptr<Stream>& IO::StreamWriter::GetStream () const [inherited]
```

get currently set stream

Get pointer to the attached stream. If there is no stream attached, an assertion will be thrown. Use HasStream() to determine if a stream is attached.

```cpp
bool IO::StreamWriter::HasStream () const [inherited]
```

return true if a stream is set

Returns true if a stream is attached to the writer.

```cpp
int Core::RefCounted::GetRefCount () const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef () [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Messaging::Port
#include <port.h>

Inheritance diagram for Messaging::Port:
Detailed Description

A message port is a receiving point for messages. Messages processed immediately and the port will be blocked until the message has been processed.

Messages are processed by message handlers which are attached to the port. More than one message handler can be attached to a port. When a message should be attached, each message handler is called in their attachment order until one of the handlers returns true, which means that the message has been handled.

For an asynchronous port implementation, which runs the message handlers in a separate thread, please check Message::AsyncPort.

(C) 2006 RadonLabs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <code>SetupAcceptedMessages ()</code></td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td><code>void AttachHandler (const Ptr&lt; Handler &gt; &amp;h)</code></td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td><code>void RemoveHandler (const Ptr&lt; Handler &gt; &amp;h)</code></td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td><code>void RemoveAllHandlers ()</code></td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td><code>SizeT GetNumHandlers () const</code></td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td><code>const Ptr&lt; Handler &gt; &amp; GetHandlerAtIndex (IndexT i) const</code></td>
<td>get a message handler by index</td>
</tr>
<tr>
<td>virtual void <code>Send (const Ptr&lt; Message &gt; &amp;msg)</code></td>
<td>send a message to the port</td>
</tr>
<tr>
<td><code>const Util::Array&lt; const Id * &gt; &amp; GetAcceptedMessages () const</code></td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td>bool <code>AcceptsMessage (const Id &amp;msgId) const</code></td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td>virtual void <code>HandleMessage (const Ptr&lt; Messaging::Message &gt; &amp;msg)</code></td>
<td>handle a single accepted message</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::String &amp;className) const</code></td>
<td></td>
</tr>
</tbody>
</table>

---
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

void RegisterMessage (const Id &msgId)

  register a single accepted message
Member Function Documentation

```cpp
void Messaging::Port::AttachHandler(const Ptr<Handler> &h)
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler(const Ptr<Handler> &h)
```

remove a message handler from the port

Remove a message handler from the port.

```cpp
void Messaging::Port::Send(const Ptr<Message> &msg) [virtual]
```

send a message to the port

Send a message to the port. This will immediately call the 
`HandleMessage()` method of all attached handlers. If the message 
has been handled by at least one of the handlers, the Handled() flag 
of the message will be set to true.

```cpp
void Messaging::Port::HandleMessage(const Ptr<Messaging::Message> &msg) [virtual]
```

handle a single accepted message

Handle a specific message. Overwrite this method in a subclass. It is 
guaranteed that this method will only be called for messages which 
are accepted by AcceptMessage().

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.
**Util::FourCC**

Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
global
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Messaging::RunThroughHandlerThread
#include <runthroughhandlerthread.h>

Inheritance diagram for Messaging::RunThroughHandlerThread:

- Core::RefCounted
- Win360::Win360Thread
- Threading::Thread
- Messaging::HandlerThreadBase
- Messaging::RunThroughHandlerThread
Detailed Description

A simple handler thread class which "runs thru", and doesn't wait for messages. This is the "old behaviour" of the N3 render thread.

(C) 2009 Radon Labs GmbH
Public Types

enum Priority

thread priorities
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RunThroughHandlerThread()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>virtual void AddMessage(const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>Add a message to be handled (override in subclass!)</td>
</tr>
<tr>
<td><code>virtual void CancelMessage(const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>Cancel a pending message (override in subclass!)</td>
</tr>
<tr>
<td><code>virtual void DoWork()</code></td>
<td>This method runs in the thread context</td>
</tr>
<tr>
<td><code>void AttachHandler(const Ptr&lt;Handler&gt; &amp;h)</code></td>
<td>Attach a message handler</td>
</tr>
<tr>
<td><code>void RemoveHandler(const Ptr&lt;Handler&gt; &amp;h)</code></td>
<td>Dynamically remove a message handler</td>
</tr>
<tr>
<td><code>void ClearHandlers()</code></td>
<td>Clear all attached message handlers</td>
</tr>
<tr>
<td><code>void WaitForHandlersOpened()</code></td>
<td>Wait until handlers have been opened</td>
</tr>
<tr>
<td><code>virtual void WaitForMessage(const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>Wait for message to be handled (optionally override in subclass!)</td>
</tr>
<tr>
<td><code>void SetPriority(Priority p)</code></td>
<td>Set the thread priority</td>
</tr>
<tr>
<td><code>Priority GetPriority()</code></td>
<td>Get the thread priority</td>
</tr>
<tr>
<td><code>void SetCoreId(System::Cpu::CoreId coreId)</code></td>
<td>Set cpu core on which the thread should be running</td>
</tr>
<tr>
<td><code>System::Cpu::CoreId GetCoreId()</code></td>
<td>Get cpu core on which the thread should be running</td>
</tr>
<tr>
<td><code>void SetStackSize(SizeT s)</code></td>
<td>Set stack size in bytes (default is 4 KByte)</td>
</tr>
<tr>
<td><code>SizeT GetStackSize()</code></td>
<td>Get stack size</td>
</tr>
<tr>
<td><code>void SetName(const Util::String &amp;n)</code></td>
<td>Set thread name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetName()</code></td>
<td>Get thread name</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Start ()</code></td>
<td>start executing the thread code, returns when thread has actually started</td>
</tr>
<tr>
<td><code>Stop ()</code></td>
<td>request threading code to stop, returns when thread has actually finished</td>
</tr>
<tr>
<td><code>IsRunning () const</code></td>
<td>return true if thread has been started</td>
</tr>
<tr>
<td><code>GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA (const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void YieldThread ()</code></td>
<td>yield the thread (gives up current time slice)</td>
</tr>
<tr>
<td><code>static void SetMyThreadName (const char *n)</code></td>
<td>set thread name from within thread context</td>
</tr>
<tr>
<td><code>static const char * GetMyThreadName ()</code></td>
<td>obtain name of thread from within thread context</td>
</tr>
<tr>
<td><code>static Threading::ThreadId GetMyThreadId ()</code></td>
<td>get the thread ID of this thread</td>
</tr>
<tr>
<td><code>static void DumpRefCountingLeaks ()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void ThreadOpenHandlers ()</strong></td>
<td>open message handlers</td>
</tr>
<tr>
<td><strong>void ThreadCloseHandlers ()</strong></td>
<td>close message handlers</td>
</tr>
<tr>
<td><strong>void ThreadUpdateHandlers ()</strong></td>
<td>open dynamically added handlers, and call <code>DoWork()</code> on all attached handlers</td>
</tr>
<tr>
<td><strong>bool ThreadUpdateDeferredMessages ()</strong></td>
<td>update deferred messages, return true if at least one message has been handled</td>
</tr>
<tr>
<td><strong>void ThreadDiscardDeferredMessages ()</strong></td>
<td>clear leftover deferred messages</td>
</tr>
<tr>
<td><strong>bool ThreadHandleMessages (const <code>Util::Array&lt; Ptr&lt; Message &gt; &gt; &amp;msgArray)</code></strong></td>
<td>handle messages in array, return true if at least one message has been handled</td>
</tr>
<tr>
<td><strong>bool ThreadHandleSingleMessage (const <code>Ptr&lt; Message &gt; &amp;msg)</code></strong></td>
<td>handle a single message without deferred support, return if message has been handled</td>
</tr>
<tr>
<td><strong>void ThreadSignalMessageHandled ()</strong></td>
<td>signal message handled event (only call if at least one message has been handled)</td>
</tr>
<tr>
<td><strong>virtual void EmitWakeupSignal ()</strong></td>
<td>override this method if your thread loop needs a wakeup call before stopping</td>
</tr>
<tr>
<td><strong>bool ThreadStopRequested () const</strong></td>
<td>check if stop is requested, call from <code>DoWork()</code> to see if the thread proc should quit</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Messaging::RunThroughHandlerThread::AddMessage(const Ptr<Message> &msg) [virtual]

add a message to be handled (override in subclass!)

This adds a new message to the thread's message queue.

Reimplemented from Messaging::HandlerThreadBase.

void Messaging::RunThroughHandlerThread::CancelMessage(const Ptr<Message> &msg) [virtual]

cancel a pending message (override in subclass!)

This removes a message from the thread's message queue, regardless of its state.

Reimplemented from Messaging::HandlerThreadBase.

void Messaging::RunThroughHandlerThread::DoWork() [virtual]

this method runs in the thread context

The message processing loop.

Reimplemented from Win360::Win360Thread.

void Messaging::HandlerThreadBase::AttachHandler(const Ptr<Handler> &h) [inherited]

attach a message handler

Attach a message handler to the port. This method may be called from
any thread.

    void Messaging::HandlerThreadBase::RemoveHandler(const Ptr<Handler> &h) [inherited]

dynamically remove a message handler

Remove a message handler. This method may be called form any thread.

    void Messaging::HandlerThreadBase::ClearHandlers() [inherited]

clear all attached message handlers

This clears all attached message handlers.

    void Messaging::HandlerThreadBase::WaitForHandlersOpened() [inherited]

wait until handlers have been opened

Wait on the handlers-opened event (will be signalled by the ThreadOpenHandlers method.

    void Messaging::HandlerThreadBase::WaitForMessage(const Ptr<Message> &msg) [virtual, inherited]

wait for message to be handled (optionally override in subclass!)

This waits until the message given as argument has been handled. In order for this message to work, the ThreadSignalMessageHandled() must be called from within the handler thread context. Note that subclasses may override this method if they need to.

Reimplemented in FrameSync::FrameSyncHandlerThread.

    void Messaging::HandlerThreadBase::ThreadOpenHandlers() [protected, inherited]
open message handlers

Open attached message handlers. This method must be called at the start of the handler thread.

```cpp
void Messaging::HandlerThreadBase::ThreadCloseHandlers() [protected, inherited]
```

close message handlers

Close attached message handlers. This method must be called right before the handler thread shuts down.

```cpp
void Messaging::HandlerThreadBase::ThreadUpdateHandlers() [protected, inherited]
```

open dynamically added handlers, and call DoWork() on all attached handlers

Do per-frame update of attached handlers. This will open handlers which have been added late, and call the DoWork() method on handlers from within the thread context.

```cpp
bool Messaging::HandlerThreadBase::ThreadUpdateDeferredMessages() [protected, inherited]
```

update deferred messages, return true if at least one message has been handled

This checks every message in the deferred message array whether it has been handled yet, if yes, the message's actual handled flag will be set, and the message will be removed from the deferred handled array. If at least one message has been handled, the method will return true, if no message has been handled, the method returns false. If message have been handled, don't forget to call ThreadSignalMessageHandled() later!

```cpp
void Messaging::HandlerThreadBase::ThreadDiscardDeferredMessages() [protected, inherited]
```

clear leftover deferred messages
This clears any leftover deferred messages. Call right before shutdown of the handler thread.

```cpp
bool Messaging::HandlerThreadBase::ThreadHandleMessages(const Util::Array<Ptr<Message>>& msgArray) [protected, inherited]

handle messages in array, return true if at least one message has been handled

Handle all message in the provided message array. Supports batched and deferred messages. Calls `ThreadHandleSingleMessage()`. If at least one message has been handled, the method returns true.

```cpp
bool Messaging::HandlerThreadBase::ThreadHandleSingleMessage(const Ptr<Message>& msg) [protected, inherited]

handle a single message without deferred support, return if message has been handled

Handle a single message, called by `ThreadHandleMessages()`. Return true if message has been handled. This method MUST be called from `ThreadHandleMessages()`, since this method will not explicitly take the handlers array critical section.

```cpp
void Messaging::HandlerThreadBase::ThreadSignalMessageHandled() [protected, inherited]

signal message handled event (only call if at least one message has been handled)

Signal the message-handled flag. Call this method once per handler-loop if either ThreadUpdateDeferredMessages or ThreadHandleMessages returns true!

```cpp
void Win360::Win360Thread::SetName(const Util::String& n) [inline, inherited]

```
set thread name

Set the thread's name. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

```cpp
const Util::String & Win360::Win360Thread::GetName() const [inline, inherited]
```

get thread name

Get the thread's name. This is the vanilla method which returns the name member. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

```cpp
void Win360::Win360Thread::Start() [inherited]
```

start executing the thread code, returns when thread has actually started

Start the thread, this creates a Win32 thread and calls the static ThreadProc, which in turn calls the virtual `DoWork()` class of this object. The method waits for the thread to start and then returns.

```cpp
void Win360::Win360Thread::Stop() [inherited]
```

request threading code to stop, returns when thread has actually finished

This stops the thread by signalling the stopRequestEvent and waits for the thread to actually quit. If the thread code runs in a loop it should use the IsStopRequested() method to see if the thread object wants it to shutdown. If so `DoWork()` should simply return.

Reimplemented in `Jobs::TPWorkerThread`.

```cpp
bool Win360::Win360Thread::IsRunning() const [inherited]
```
return true if thread has been started

Returns true if the thread is currently running.

```cpp
void
Win360::Win360Thread::YieldThread() [static, inherited]
```

yield the thread (gives up current time slice)

The yield function is empty on Win32 and Xbox360.

```cpp
void
Win360::Win360Thread::SetMyThreadName(const char * n) [static, inherited]
```

set thread name from within thread context

Static method which sets the name of this thread. This is called from within ThreadProc. The string pointed to must remain valid until the thread is terminated!

```cpp
const char *
Win360::Win360Thread::GetMyThreadName() [static, inherited]
```

obtain name of thread from within thread context

Static method to obtain the current thread name from anywhere in the thread's code.

```cpp
Threading::ThreadId
Win360::Win360Thread::GetMyThreadId() [static, inherited]
```

get the thread ID of this thread

Static method which returns the ThreadId of this thread.

```cpp
void
Win360::Win360Thread::EmitWakeupSignal() [protected, virtual, inherited]
```

override this method if your thread loop needs a wakeup call before stopping
This method is called by `Thread::Stop()` after setting the stopRequest event and before waiting for the thread to stop. If your thread runs a loop and waits for jobs it may need an extra wakeup signal to stop waiting and check for the `ThreadStopRequested()` event. In this case, override this method and signal your event object.

Reimplemented in `Jobs::TPWorkerThread`, and `Messaging::BlockingHandlerThread`.

```cpp
bool Win360::Win360Thread::ThreadStopRequested() const [inline, protected, inherited]
```

Check if stop is requested, call from `DoWork()` to see if the thread proc should quit.

If the derived `DoWork()` method is running in a loop it must regularly check if the process wants the thread to terminate by calling `ThreadStopRequested()` and simply return if the result is true. This will cause the thread to shut down.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

Get the current refcount.

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

Decrement refcount and destroy object if refcount is zero.

Decrement the refcount and destroy object if refcount is zero.
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Messaging::StaticMessageHandler
Messaging::StaticMessageHandler
Class Reference

#include <staticmessagehandler.h>
Detailed Description

Implements a simple, static message handler helper class. This separates the tedious message handling code into a separate class, so that the main class doesn't have to be polluted with message handling code.

(C) 2010 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Template</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>template&lt;class OBJTYPE&gt;</td>
<td>static void</td>
<td><strong>Dispatch</strong> (const <strong>Ptr</strong>&lt; OBJTYPE &gt; &amp;object, const <strong>Ptr</strong>&lt; Message &gt; &amp;msg) dispatch a message to handling method</td>
</tr>
<tr>
<td>template&lt;class OBJTYPE, class MSGTYPE&gt;</td>
<td>static void</td>
<td><strong>Handle</strong> (const <strong>Ptr</strong>&lt; OBJTYPE &gt; &amp;object, const <strong>Ptr</strong>&lt; MSGTYPE &gt; &amp;msg) a handler method with object association</td>
</tr>
<tr>
<td>template&lt;class MSGTYPE&gt;</td>
<td>static void</td>
<td><strong>Handle</strong> (const <strong>Ptr</strong>&lt; MSGTYPE &gt; &amp;msg) a handler method without object association</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](http://www.doxygen.org) at Fri Mar 26 15:21:48 2010
MGraphics::VisibilityBox
MGraphics::VisibilityBox Class Reference

#include <visibilitybox.h>
Detailed Description

A VisibilityBox implements an transformed bounding box which groups graphics entities into a visibility group.

(C) 2010 Radon Labs GmbH
MGraphics::VisibilityCluster
MGraphics::VisibilityCluster Class Reference

#include <visibilitycluster.h>
Detailed Description

Culls the attached graphics entities if the viewer is inside a bounding box cluster. VisibilityClusters are created and manually configured by level designers inside the level editor.

(C) 2010 Radon Labs GmbH
Models::ManagedModel
#include <managedmodel.h>

Inheritance diagram for Models::ManagedModel:
Detailed Description

A specialized managed resource for Models.

(C) 2007 Radon Labs GmbH
Public Types

enum Priority

priority levels
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const Ptr&lt; Model &gt; &amp; GetModel () const</code></td>
<td>get contained model resource</td>
</tr>
<tr>
<td><code>void ClearRenderStats ()</code></td>
<td>clear render statistics</td>
</tr>
<tr>
<td><code>void UpdateRenderStats (float lod)</code></td>
<td>update render statistics (called by client)</td>
</tr>
<tr>
<td><code>void UpdateRenderStats (const Math::float2 &amp;screenSpaceSize)</code></td>
<td>update render statistics (called by client)</td>
</tr>
<tr>
<td><code>void SetResourceId (const ResourceId &amp;id)</code></td>
<td>set resource id</td>
</tr>
<tr>
<td><code>const ResourceId &amp; GetResourceId () const</code></td>
<td>get resource id</td>
</tr>
<tr>
<td><code>void SetResourceType (const Core::Rtti *rtti)</code></td>
<td>set contained resource type</td>
</tr>
<tr>
<td><code>const Core::Rtti * GetResourceType () const</code></td>
<td>get contained resource type</td>
</tr>
<tr>
<td><code>void IncrClientCount ()</code></td>
<td>increment client count</td>
</tr>
<tr>
<td><code>void DecrClientCount ()</code></td>
<td>decrement client count</td>
</tr>
<tr>
<td><code>SizeT GetClientCount () const</code></td>
<td>get current client count</td>
</tr>
<tr>
<td><code>SizeT GetRenderCount () const</code></td>
<td>get render count for this frame (number of calls to UpdateRenderStats())</td>
</tr>
<tr>
<td><code>float GetResourceStreamingLevelOfDetail () const</code></td>
<td>get resourceStreamingLevelOfDetail for this frame</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Resource::State::GetState() const</td>
<td>get the current priority</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetLoadedResource() const</td>
<td>get current resource loading state (Initial, Pending, Loaded, Failed, Cancelled)</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetResource() const</td>
<td>get contained resource or placeholder if resource is invalid or not loaded</td>
</tr>
<tr>
<td>bool IsPlaceholder() const</td>
<td>return true if the placeholder resource would be returned</td>
</tr>
<tr>
<td>void Clear()</td>
<td>clear the contained resource</td>
</tr>
<tr>
<td>bool IsAutoManaged() const</td>
<td>returns true if autoManaged</td>
</tr>
<tr>
<td>void SetAutoManaged(const bool autoManaged)</td>
<td>sets autoManaged-flag</td>
</tr>
<tr>
<td>IndexT GetLastFrameId() const</td>
<td>returns frameld this resource was latest referenced to</td>
</tr>
<tr>
<td>void SetFrameId(const SizeT frameld)</td>
<td>sets the frameld the resource was latest used at</td>
</tr>
<tr>
<td>void SetResource(const Ptr&lt; Resource &gt; &amp;resource)</td>
<td>set actual resource</td>
</tr>
<tr>
<td>void SetPlaceholder(const Ptr&lt; Resource &gt; &amp;placeholder)</td>
<td>set placeholder resource</td>
</tr>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>return true if this object is instance of given class by string</code></td>
<td><code>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
</tr>
<tr>
<td><code>return true if this object is instance of given class by fourcc</code></td>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
</tr>
<tr>
<td><code>return true if this object is instance of given class, or a derived class</code></td>
<td><code>IsA (const Util::String &amp;rttiName) const</code></td>
</tr>
<tr>
<td><code>return true if this object is instance of given class, or a derived class, by string</code></td>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
</tr>
<tr>
<td><code>get the class name</code></td>
<td><code>const Util::String &amp; GetClassName () const</code></td>
</tr>
<tr>
<td><code>get the class FourCC code</code></td>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DumpRefCountingLeaks()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Resources::ManagedResource::ClearRenderStats() [inherited]
clear render statistics

This method resets the current render stats and is usually called during the Prepare() method of the ResourceManager (before rendering is started for the current frame).

void Resources::ManagedResource::UpdateRenderStats(float lod) [inherited]
update render statistics (called by client)

This method is called by the resource client during rendering to write back render statistics. If the resource isn't rendered, the method MUST NOT be called. If the resource is rendered, the client must provide a screen space size guesstimate which will be used by the ResourceManager to bump or drop the lod of the resource.

void Resources::ManagedResource::UpdateRenderStats(const Math::float2& screenSpaceSize) [inherited]
update render statistics (called by client)

This method is called by the resource client during rendering to write back render statistics. If the resource isn't rendered, the method MUST NOT be called. If the resource is rendered, the client must provide a screen space size guesstimate which will be used by the ResourceManager to bump or drop the lod of the resource.

int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar
Models::Model
Models::Model Class Reference

#include <model.h>

Inheritance diagram for Models::Model:
Detailed Description

A Model represents the template for a renderable object, consisting of a hierarchy of ModelNodes which represent transformations and shapes. Models should generally be created through the ModelServer, which guarantees that a given Model is only loaded once into memory. To render a Model, at least one ModelInstance must be created from the Model. Usually one ModelInstance is created per game object. Generally speaking, all per-instance data lives in the ModelInstance objects, while all constant data lives in the Model object.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>State</th>
</tr>
</thead>
</table>

*resource states (DO NOT CHANGE ORDER!)*
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual</td>
<td><strong>~Model ()</strong> destructor</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Unload ()</strong> unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td>void</td>
<td><strong>LoadResources ()</strong> load node resources (meshes, textures, shaders, ...)</td>
</tr>
<tr>
<td>void</td>
<td><strong>UnloadResources ()</strong> unload node resources</td>
</tr>
<tr>
<td>bool</td>
<td><strong>CheckPendingResources ()</strong> check if all resources have been loaded, return true if yes</td>
</tr>
<tr>
<td>void</td>
<td><strong>UpdateBoundingBox ()</strong> update model's bounding box from model nodes</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetBoundingBox</strong> (const Math::bbox &amp;b) set the model's local bounding box</td>
</tr>
<tr>
<td>const Math::bbox &amp;</td>
<td><strong>GetBoundingBox ()</strong> const get the model's local bounding box</td>
</tr>
<tr>
<td>Ptr&lt;ModelNode&gt;</td>
<td><strong>LookupNode</strong> (const Util::String &amp;path) const lookup a ModelNode in the Model by path, returns invalid pointer if not found</td>
</tr>
<tr>
<td>void</td>
<td><strong>AttachNode</strong> (const Ptr&lt;ModelNode&gt; &amp;node) attach a model node to the Model</td>
</tr>
<tr>
<td>void</td>
<td><strong>RemoveNode</strong> (const Ptr&lt;ModelNode&gt; &amp;node) remove a model node from the Model</td>
</tr>
<tr>
<td>const Util::Array&lt;Ptr&lt;ModelNode&gt;&gt; &amp;</td>
<td><strong>GetNodes ()</strong> const access to model nodes</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const Ptr&lt; ModelNode &gt; &amp; GetRootNode()</code></td>
<td>get root node (always at index 0)</td>
</tr>
<tr>
<td><code>Ptr&lt; ModelInstance &gt; CreateInstance()</code></td>
<td>create a <code>ModelInstance</code> of the <code>Model</code></td>
</tr>
<tr>
<td><code>Ptr&lt; ModelInstance &gt; CreatePartialInstance(const Util::StringAtom &amp;rootNodePath, const Math::matrix44 &amp;rootNodeOffsetMatrix)</code></td>
<td>create a <code>ModelInstance</code> from a hierarchy sub-tree</td>
</tr>
<tr>
<td><code>void DiscardInstance(Ptr&lt; ModelInstance &gt; modelInstance)</code></td>
<td>discard an instance</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Ptr&lt; ModelInstance &gt; &gt; &amp; GetInstances()</code></td>
<td>get all attached model instances</td>
</tr>
<tr>
<td><code>void SetAsyncEnabled(bool b)</code></td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td><code>bool IsAsyncEnabled()</code></td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>void Lock()</code></td>
<td>set locked to true</td>
</tr>
<tr>
<td><code>void Unlock()</code></td>
<td>set locked to false</td>
</tr>
<tr>
<td><code>bool IsLocked()</code></td>
<td>returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td><code>void SetResourceld(const Resourceld &amp;id)</code></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><code>const Resourceld &amp; GetResourceld()</code></td>
<td>get the resource identifier</td>
</tr>
<tr>
<td><code>void SetLoader(const Ptr&lt;ResourceLoader&gt; &amp;loader)</code></td>
<td>set optional resource loader</td>
</tr>
</tbody>
</table>
constPtr< ResourceLoader > &GetLoader () const
get optional resource loader

void SetSaver (constPtr< ResourceSaver > &saver)
set optional resource saver

constPtr< ResourceSaver > &GetSaver () const
get optional resource saver

SizeT GetUseCount () const
get current use count

virtual State Load ()
load the resource

void SetState (State s)
set current state (usually only called during Load())!

State GetState () const
get current state

bool IsLoaded () const
return true if current state is Loaded

bool IsPending () const
return true if current state is Pending

bool LoadFailed () const
return true if current state is Failed

virtual bool Save ()
save the resource

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String className) const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsInstanceOf</code> (const <code>Util::FourCC</code> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA</code> (const <code>Util::String</code> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName</code> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC</code> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void <strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
**Protected Member Functions**

<table>
<thead>
<tr>
<th>void</th>
<th><strong>IncrUseCount</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>increment use count</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>DecrUseCount</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void Models::Model::LoadResources()

load node resources (meshes, textures, shaders, ...)

This method asks all model nodes to load their resources. Note that actual resource loading may be asynchronous and placeholder resources may be in place right after this method returns.

```cpp
void Models::Model::UnloadResources()

unload node resources

This method asks all model nodes to unload their resources.

```cpp
bool Models::Model::CheckPendingResources()

check if all resources have been loaded, return true if yes

This checks whether all resource have been loaded, if yes the method OnResourcesLoaded() will be called once. If some resources are not loaded yet, the method will return false.

```cpp
void Models::Model::UpdateBoundingBox()

update model's bounding box from model nodes

This method will update the Model's bounding box to include the bounding boxes of all ModelNodes owned by the **Model** object.

```cpp
Ptr<ModelNode> Models::Model::LookupNode(const Util::String path) const

lookup a **ModelNode** in the **Model** by path, returns invalid pointer if
not found

Careful, this method is SLOW!

```cpp
const Ptr<ModelNode> & Models::Model::GetRootNode() const [inline]
```

get root node (always at index 0)

Get a pointer to the root node. This is always the first node.

```cpp
Resource::State Resources::Resource::Load() [virtual, inherited]
```

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the `IsPending()` method will return true as long as the load is in progress, and `IsLoaded()` will become true when the loading process has finished. If the load has failed, `IsPending()` will switch to false and `IsLoaded()` will not be true.

```cpp
bool Resources::Resource::Save() [virtual, inherited]
```

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Models::ModellInstance
Models::ModellInstance Class Reference

#include <modelinstance.h>

Inheritance diagram for Models::ModellInstance:
Detailed Description

A **ModellInstance** contains the per-instance data for rendering a **Model**. Usually there is one **ModellInstance** created per game object.

A **ModellInstance** is roughly comparable to a Nebula2 nRenderContext.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ModelInstance ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~ModelInstance ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>bool <strong>IsValid ()</strong> const</td>
<td>return true if currently valid</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; Model &gt; &amp;</strong> <strong>GetModel ()</strong> const</td>
<td>get the Model this instance was created from</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; InternalGraphics::InternalModelEntity &gt; &amp;</strong> <strong>GetModelEntity ()</strong> const</td>
<td>get the ModelEntity which owns this instance</td>
</tr>
<tr>
<td>void <strong>SetTransform (const Math::matrix44 &amp;m)</strong></td>
<td>set world space transform of the instance</td>
</tr>
<tr>
<td>const <strong>Math::matrix44 &amp;</strong> <strong>GetTransform ()</strong> const</td>
<td>get world space transform</td>
</tr>
<tr>
<td><strong>Ptr&lt; ModelNodeInstance &gt;</strong> <strong>LookupNodeInstance (const Util::StringAtom &amp;path)</strong> const</td>
<td>lookup a node instance, return invalid ptr if not exists, this method is SLOW</td>
</tr>
<tr>
<td>const <strong>Util::Array&lt; Ptr&lt; ModelNodeInstance &gt; &gt; &amp;</strong> <strong>GetNodeInstances ()</strong> const</td>
<td>get all node instances</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; ModelNodeInstance &gt; &amp;</strong> <strong>GetRootNodeInstance ()</strong> const</td>
<td>get pointer to top-level node instance</td>
</tr>
<tr>
<td>void <strong>AttachNodeInstance (const Ptr&lt; ModelNodeInstance &gt; &amp;nodeInst)</strong></td>
<td>attach a node instance which has already been setup</td>
</tr>
<tr>
<td>void <strong>RemoveNodeInstance (const Ptr&lt; ModelNodeInstance &gt; &amp;nodeInst)</strong></td>
<td>remove a node instance, does not</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void OnNotifyCullingVisible (IndexT frameIndex, Timing::Time time)</code></td>
<td>called from <code>ModelEntity::OnNotifyCullingVisible</code></td>
</tr>
<tr>
<td><code>void OnRenderBefore (IndexT frameIndex, Timing::Time time)</code></td>
<td>called from <code>ModelEntity::OnRenderBefore</code></td>
</tr>
<tr>
<td><code>void OnVisibilityResolve (IndexT resolveIndex, const Math::matrix44 &amp;cameraTransform)</code></td>
<td>called during visibility resolve</td>
</tr>
<tr>
<td><code>void UpdateRenderStats (const Math::matrix44 &amp;cameraTransform)</code></td>
<td>calls <code>ModelInstance::OnVisibilityResolve</code> + updating screen space stats</td>
</tr>
<tr>
<td><code>void RenderDebug ()</code></td>
<td>render node specific debug shape</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Setup (const Ptr&lt; Model &gt; &amp;model, const Ptr&lt; ModelNode &gt; &amp;rootModelNode)</code></td>
<td>setup the <code>ModellInstance</code> from a root model node</td>
</tr>
<tr>
<td><code>Discard ()</code></td>
<td>cleanup and unlink from model</td>
</tr>
<tr>
<td><code>SetModelEntity (const Ptr&lt; InternalGraphics::InternalModelEntity &gt; &amp;mdlEntity)</code></td>
<td>set pointer to <code>ModelEntity</code> which owns this instance</td>
</tr>
<tr>
<td><code>OnShow (Timing::Time time)</code></td>
<td>called when the entity becomes visible with current time</td>
</tr>
<tr>
<td><code>OnHide (Timing::Time time)</code></td>
<td>called when the entity becomes invisible</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Ptr< ModelNodeInstance >**
Models::ModelInstance::LookupNodeInstance ( **Util::StringAtom path** ) const

lookup a node instance, return invalid ptr if not exists, this method is SLOW

Careful, this method is SLOW!

**const Ptr< ModelNodeInstance > &**
Models::ModelInstance::GetRootNodeInstance ( ) const

get pointer to top-level node instance

Get a pointer to the root node instance. This is always the first node instance.

**void**
Models::ModelInstance::AttachNodeInstance ( **ModelNodeInstance nodeInst** ) &

attach a node instance which has already been setup

Attach a node instance which has already been setup.

**void**
Models::ModelInstance::RemoveNodeInstance ( **ModelNodeInstance nodeInst** ) &

remove a node instance, does not discard the node instance!

Remove a node instance, do not discard it!

**void**
Models::ModelInstance::OnNotifyCullingVisible ( **IndexT frameIndex**, **Timing::Time time** )

called from ModelEntity::OnNotifyCullingVisible
This method is called once per frame on ModelInstances which have been detected as visible during the culling phase (so, relatively early).

```
void Models::ModelInstance::OnRenderBefore ( IndexT frameIndex,
                                           Timing::Time time
 )
```
called from ModelEntity::OnRenderBefore

This method is called once per frame on visible ModelInstances right before rendering.

```
void Models::ModelInstance::OnVisibilityResolve ( IndexT resolveIndex,
                                                   const Math::matrix44& cameraTransform
 )
```
called during visibility resolve

This method is called by the **Graphics** subsystem when we are currently visible. Once all visible model instances are notified, the **Graphics** subsystem can get a render-order-optimized list of all visible model-node-instances through the **ModelServer**.

```
void Models::ModelInstance::UpdateRenderStats ( const Math::matrix44& cameraTransform
 )
```
calls **ModelInstance::OnVisibilityResolve(...) + updating screen space stats**

This updates the render stats of the model. This may only be called by the players camera as other camera transforms (e.g. for light and shadow) shall not influence the LOD of a model.

```
void Models::ModelInstance::RenderDebug ( )
```
render node specific debug shape
This method is called from the RenderDebug of Graphics::ModelEntity.

```cpp
int Core::RefCounted::GetRefCount( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Models::ModelNode
#include <modelnode.h>

Inheritance diagram for Models::ModelNode:
Detailed Description

Represents a transformation hierarchy element inside a Model. Subclasses of ModelNodes represent transformations and geometry of a 3D model arranged in 3d hierarchy (but not in a logical hierarchy of C++ object, instead model nodes live in a flat array to prevent recursive iteration).

A ModelNode is roughly comparable to a nSceneNode in Nebula2.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ModelNode ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~ModelNode ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt; ModelNodeInstance &gt; CreateNodeInstance ()</code> const</td>
<td>create a model node instance</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt; ModelNodeInstance &gt; CreateNodeInstanceHierarchy (Ptr&lt; ModellInstance &gt; &amp;model)</code></td>
<td>recursively create model node instance and model node instances</td>
</tr>
<tr>
<td>virtual void <code>OnAttachToModel (const Ptr&lt; I &gt; &amp;model)</code></td>
<td>called when attached to model node</td>
</tr>
<tr>
<td>virtual void <code>OnRemoveFromModel ()</code></td>
<td>called when removed from model node</td>
</tr>
<tr>
<td>virtual void <code>LoadResources ()</code></td>
<td>called when resources should be loaded</td>
</tr>
<tr>
<td>virtual void <code>UnloadResources ()</code></td>
<td>called when resources should be unloaded</td>
</tr>
<tr>
<td>virtual void <code>OnResourcesLoaded ()</code></td>
<td>called once when all pending resource have been loaded</td>
</tr>
<tr>
<td>virtual void <code>BeginParseDataTags ()</code></td>
<td>begin parsing data tags</td>
</tr>
<tr>
<td>virtual bool <code>ParseDataTag (const Util::FourCC &amp;fourCC, const Ptr&lt; IO::BinaryReader &gt; &amp;reader)</code></td>
<td>parse data tag (called by loader code)</td>
</tr>
<tr>
<td>virtual void <code>EndParseDataTags ()</code></td>
<td>finish parsing data tags</td>
</tr>
<tr>
<td>virtual void <code>ApplySharedState (IndexT frameIndex)</code></td>
<td>apply state shared by all my ModelNodeInstances</td>
</tr>
<tr>
<td>virtual <code>Resources::Resource::State GetResourceState ()</code> const</td>
<td>get resource state</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsAttachedToModel()</code> const</td>
<td>return true if currently attached to a Model</td>
</tr>
<tr>
<td><code>const Ptr&lt; Model &gt; &amp; GetModel()</code> const</td>
<td>get model this node is attached to</td>
</tr>
<tr>
<td><code>void SetResourceStreamingLevelOfDetail(float factor)</code></td>
<td>sets the resourceStreamingLevelOfDetail but only if the given value is bigger than the current one (reseted on frame-start)</td>
</tr>
<tr>
<td><code>void ResetScreenSpaceStats()</code></td>
<td>resets all screen space stats e.g. size</td>
</tr>
<tr>
<td><code>void SetBoundingBox(const Math::bbox &amp;b)</code></td>
<td>set bounding box</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetBoundingBox()</code> const</td>
<td>get bounding box of model node</td>
</tr>
<tr>
<td><code>void SetName(const Util::StringAtom &amp;)</code></td>
<td>set model node name</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetName()</code> const</td>
<td>get model node name</td>
</tr>
<tr>
<td><code>void SetType(const ModelNodeType::Code)</code></td>
<td>set <code>ModelNodeType</code></td>
</tr>
<tr>
<td><code>ModelNodeType::Code GetTyp()</code> const</td>
<td>get the <code>ModelNodeType</code></td>
</tr>
<tr>
<td><code>void SetParent(const Ptr&lt; ModelNode &gt; &amp;p)</code></td>
<td>set parent node</td>
</tr>
<tr>
<td><code>const Ptr&lt; ModelNode &gt; &amp; GetParent()</code> const</td>
<td>get parent node</td>
</tr>
<tr>
<td><code>bool HasParent()</code> const</td>
<td>return true if node has a parent</td>
</tr>
<tr>
<td><code>void AddChild(const Ptr&lt; ModelNode &gt; &amp;c)</code></td>
<td>add a child node</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Ptr&lt; ModelNode &gt; &gt; &amp; GetChildren()</code> const</td>
<td>get child nodes</td>
</tr>
</tbody>
</table>
bool HasChild (const Util::StringAtom &name) const
return true if a direct child exists by name

const Ptr<ModelNode> & LookupChild (const Util::StringAtom &name) const
get pointer to direct child by name

void AddVisibleNodeInstance (IndexT frameIndex, const Ptr<ModelNodeInstance> &nodeInst)
called by model node instance on NotifyVisible()

const Util::Array< Ptr<ModelNodeInstance> > & GetVisibleModelNodeInstances (ModelNodeType::Code t) const
get visible model node instances

bool HasStringAttr (const Util::StringAtom &attrId) const
has string attr

const Util::StringAtom & GetStringAttr (const Util::StringAtom &attrId) const
get string attr

void SetStringAttr (const Util::StringAtom &attrId, const Util::StringAtom &value)
add string attribute

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
<table>
<thead>
<tr>
<th>Function Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class or a derived class.</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class or a derived class, by string.</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class or a derived class, by fourcc.</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
<tr>
<td>Protected Member Functions</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt; ModelNodeInstance &gt;</code> &amp;modelInst, const <code>Ptr&lt; ModelInstance &gt;</code> &amp;modelInst, const <code>Ptr&lt; ModelNodeInstance &gt;</code> &amp;parentNodeInst)</td>
</tr>
<tr>
<td><code>RecurseCreateNodeInstanceHierarchy</code></td>
</tr>
<tr>
<td>recursively create node instance hierarchy</td>
</tr>
</tbody>
</table>
## Protected Attributes

<table>
<thead>
<tr>
<th>float</th>
<th><code>resourceStreamingLevelOfDetail</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>factor between 0.0 (close) and 1.0 (far away) describing the distance to camera (used for decision of max needed mipMap)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
Ptr< ModelNodeInstance >
Models::ModelNode::CreateNodeInstanceHierarchy(
    const Ptr< ModelInstance >& modelInst)
```

recursively create model node instance and child model node instances

Create the node instance hierarchy.

```cpp
void
Models::ModelNode::LoadResources()
```[virtual]

called when resources should be loaded

This method is called when required resources should be loaded.

Reimplemented in Characters::CharacterNode,
Models::ShapeNode, Models::StateNode, and
Particles::ParticleSystemNode.

```cpp
void
Models::ModelNode::UnloadResources()
```[virtual]

called when resources should be unloaded

This method is called when required resources should be unloaded.

Reimplemented in Characters::CharacterNode,
Models::ShapeNode, Models::StateNode, and
Particles::ParticleSystemNode.

```cpp
void
Models::ModelNode::OnResourcesLoaded()
```[virtual]

called once when all pending resource have been loaded

This method is called once by Model::OnResourcesLoaded() when all pending resources of a model have been loaded.
Reimplemented in **Characters::CharacterNode**, and **Particles::ParticleSystemNode**.

```cpp
void Models::ModelNode::BeginParseDataTags() [virtual]

begin parsing data tags

Begin parsing data tags. This method is called by **StreamModelLoader** before **ParseDataTag()** is called for the first time.

```cpp
bool Models::ModelNode::ParseDataTag(const Util::FourCC & fourCC, const Ptr<IO::BinaryReader> & reader) [virtual]

parse data tag (called by loader code)

Parse a single data tag. If a subclass doesn't accept the data tag, the parent class method must be called!

Reimplemented in **Characters::CharacterNode**, **Characters::CharacterSkinNode**, **Models::ShapeNode**, **Models::StateNode**, **Models::TransformNode**, and **Particles::ParticleSystemNode**.

```cpp
void Models::ModelNode::EndParseDataTags() [virtual]

finish parsing data tags

End parsing data tags. This method is called by **StreamModelLoader** after the last **ParseDataTag()** is called.

```cpp
void Models::ModelNode::ApplySharedState(IndexT frameIndex) [virtual]

apply state shared by all my ModelNodeInstances
This method is called once before rendering the ModelNode's visible instance nodes through the `ModelNodeInstance::ApplyState()` and `ModelNodeInstance::Render()` methods. The method must apply the shader state that is shared across all instances. Since this state is constant across all instance nodes, this only happens once before rendering an instance set.

Reimplemented in `Characters::CharacterSkinNode`, `Models::ShapeNode`, and `Models::StateNode`.

```cpp
Resource::State
Models::ModelNode::GetResourceState( ) const [virtual]
```

get overall state of contained resources (Initial, Loaded, Pending, Failed, Cancelled)

Returns the overall resource state (Initial, Loaded, Pending, Failed, Cancelled). Higher states override lower state (if some resources are already Loaded, and some are Pending, then Pending will be returned, if some resources failed to load, then Failed will be returned, etc...). Subclasses which load resource must override this method, and modify the return value of the parent class as needed).

Reimplemented in `Characters::CharacterNode`, `Models::ShapeNode`, `Models::StateNode`, `Models::TransformNode`, and `Particles::ParticleSystemNode`.

```cpp
void
Models::ModelNode::ResetScreenSpaceStats( ) [inline]
```

resets all screen space stats e.g. size

Reset resourceStreamingLevelOfDetail to -1.0 as we are able to recognize invisible items this way. (visible items will overwrite this value with a value >= 0.0)

```cpp
Ptr< ModelNodeInstance >
Models::ModelNode::RecurseCreateNodeInstanceHierarchy( const Ptr< ModelInstance > & modelInst, const Ptr< ModelNodeInstance > parentNodeInst >&)
```
recursively create node instance hierarchy

Recursively create node instances and attach them to the provided model instance. Returns a pointer to the root node instance.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String& Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks()

[dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)]

This method should be called as the very last before an application exits.
Models::ModelNodeInstance
Models::ModelNodeInstance Class Reference

#include <modelnodeinstance.h>

Inheritance diagram for Models::ModelNodeInstance:

```
Core::RefCounted
  ↓
Models::ModelNodeInstance
  ↓
Models::TransformNodeInstance
  ↓
Characters::CharacterNodeInstance  Models::StateNodeInstance
  ↓
Models::ShapeNodeInstance  Particles::ParticleSystemNodeInstance
  ↓
Characters::CharacterSkinNodeInstance
```
Detailed Description

A ModelNodeInstance holds the per-instance data of a ModelNode and does most of the actually interesting Model rendering stuff.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ModelNodeInstance ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~ModelNodeInstance ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <strong>Setup</strong> (const Ptr&lt;ModelInstance&gt; &amp;inst, const Ptr&lt;ModelNode&gt; &amp;node, const Ptr&lt;ModelNodeInstance&gt; &amp;parentNodeInst)</td>
<td>setup the model node instance</td>
</tr>
<tr>
<td>virtual void <strong>Discard</strong> ()</td>
<td>discard the model node instance</td>
</tr>
<tr>
<td>virtual void <strong>DiscardHierarchy</strong> ()</td>
<td>discard the model node instance and all of its children</td>
</tr>
<tr>
<td>bool <strong>IsValid () const</strong></td>
<td>return true if the model node instance is valid</td>
</tr>
<tr>
<td>virtual void <strong>OnNotifyCullingVisible</strong> (IndexT frameIndex, Timing::Time time)</td>
<td>called from ModelEntity::OnNotifyCullingVisible</td>
</tr>
<tr>
<td>virtual void <strong>OnRenderBefore</strong> (IndexT frameIndex, Timing::Time time)</td>
<td>called from ModelEntity::OnRenderBefore</td>
</tr>
<tr>
<td>virtual void <strong>OnVisibilityResolve</strong> (IndexT resolveIndex, float distanceToViewer)</td>
<td>called during visibility resolve</td>
</tr>
<tr>
<td>virtual void <strong>ApplyState</strong> ()</td>
<td>apply per-instance state prior to</td>
</tr>
</tbody>
</table>
virtual void Render ()
perform rendering

const Util::StringAtom & GetName () const
get model node name

bool HasParent () const
return true if node has a parent

const Ptr<ModelNodeInstance> & GetParent () const
get parent node

const Util::Array< Ptr<ModelNodeInstance> > & GetChildren () const
get child nodes

bool HasChild (const Util::StringAtom &name) const
return true if a direct child exists by name

const Ptr<ModelNodeInstance> & LookupChild (const Util::StringAtom &name) const
get pointer to direct child by name

Ptr<ModelNodeInstance> LookupPath (const Util::String &path)
get modelnodeinstance by hierarchy path

const Ptr<ModelInstance> & GetModellInstance () const
get the ModellInstance we are attached to

const Ptr<ModelNode> & GetModelNode () const
get the ModelNode we’re associated with

void SetVisible (bool b, Timing::Time time, bool recursive=true)
set the node instance’s visibility

bool IsVisible () const
return true if node instance is set to visible

IndexT GetModelNodeInstanceIndex () const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA(const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA(const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName() const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC() const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump reference leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>SetParent</strong></td>
<td>(const Ptr&lt; ModelNodeInstance &gt; &amp;p) set parent node</td>
</tr>
<tr>
<td>void</td>
<td><strong>AddChild</strong></td>
<td>(const Ptr&lt; ModelNodeInstance &gt; &amp;c) add a child node</td>
</tr>
<tr>
<td>virtual</td>
<td><strong>RenderDebug</strong></td>
<td>() render node specific debug shape</td>
</tr>
<tr>
<td>virtual</td>
<td><strong>OnShow</strong></td>
<td>(Timing::Time time) called when the node becomes visible with current time</td>
</tr>
<tr>
<td>virtual</td>
<td><strong>OnHide</strong></td>
<td>(Timing::Time time) called when the node becomes invisible</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::ModelNodeInstance::DiscardHierarchy( ) [virtual]

discard the model node instance and all of its children

Discards this model node instance and all of its children recursively.

void Models::ModelNodeInstance::ApplyState( ) [virtual]

apply per-instance state prior to rendering

The **ApplyState()** method is called when per-instance shader-state should be applied. This method may be called several times per frame. The calling order is always in "render order", first, the **ApplyState()** method on the **ModelNode** will be called, then each **ApplyState()** and **Render()** method of the **ModelNodeInstance** objects.

Reimplemented in **Models::StateNodeInstance**, **Models::TransformNodeInstance**, and **Particles::ParticleSystemNodeInstance**.

void Models::ModelNodeInstance::Render( ) [virtual]

perform rendering

The **Render()** method is called when the **ModelNodeInstance** needs to render itself. There will always be a call to the Apply() method before **Render()** is called, however, **Render()** may be called several times per Apply() invokation.

Reimplemented in **Characters::CharacterSkinNodeInstance**, **Models::ShapeNodeInstance**, and **Particles::ParticleSystemNodeInstance**.
bool ( ) const Models::ModelNodeInstance::IsVisible

return true if node instance is set to visible

FIXME: The recursion in this is method makes it slow, especially in deep hierarchies. Also it can't be inlined.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Models::ModelNodeType
Models::ModelNodeType Class Reference

#include <modelnodetype.h>
Detailed Description

ModelNodeTypes identify a ModelNode for a specific rendering pass. Works the same as shader features, there is no hardcoded set of ModelNodeTypes, but there is a central registry which converts string into binary indices, and guarantees that the returned indices for a name are the same for the lifetime of the application.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>typedef <code>Util::StringAtom</code></td>
<td>human readable name of a <code>ModelNodeType</code></td>
<td>binary code for a <code>ModelNodeType</code></td>
</tr>
<tr>
<td>typedef <code>IndexT</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```cpp
typedef Util::StringAtom Name
    human readable name of a ModelNodeType

typedef IndexT Code
    binary code for a ModelNodeType
```
## Static Public Member Functions

| static `Code` FromName (const `Name` &name) | convert from string |
| static `Name` ToName (Code c) | convert to string |
### Static Public Attributes

<table>
<thead>
<tr>
<th>static const IndexT</th>
<th>MaxNumModelNodeTypes = 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>maximum number of different ModelNodeTypes</td>
</tr>
<tr>
<td>static const IndexT</td>
<td>InvalidModelNodeType = InvalidIndex</td>
</tr>
<tr>
<td></td>
<td>invalid model node type code</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:48 2010
Models::ModelServer
Models::ModelServer Class Reference

#include <modelserver.h>

Inheritance diagram for Models::ModelServer:

```
[Diagram showing inheritance relationship between Core::RefCounted and Models::ModelServer]
```
Detailed Description

The ModelServer loads and creates shared Model objects.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ModelServer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~ModelServer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>SetModelResourceMapper (const Resources::ResourceMapper &amp;mapper)</strong></td>
<td>set resource mapper for creating models</td>
</tr>
<tr>
<td><strong>GetModelResourceMapper () const</strong></td>
<td>get resource mapper</td>
</tr>
<tr>
<td><strong>Open ()</strong></td>
<td>open the model server</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>close the model server</td>
</tr>
<tr>
<td><strong>IsOpen () const</strong></td>
<td>return true if model server is open</td>
</tr>
<tr>
<td><strong>HasManagedModel (const Resources::ResourceId &amp;resId) const</strong></td>
<td>check if a managed model exists</td>
</tr>
<tr>
<td><strong>LoadManagedModel (const Resources::ResourceId &amp;resId)</strong></td>
<td>load a managed Model from URI</td>
</tr>
<tr>
<td><strong>LookupManagedModel (const Resources::ResourceId &amp;resId) const</strong></td>
<td>lookup an existing model</td>
</tr>
<tr>
<td><strong>DiscardManagedModel (const ManagedModel &amp;managedModel)</strong></td>
<td>discard a managed model</td>
</tr>
<tr>
<td><strong>ConsumeNewModelNodeInstanceIndex ()</strong></td>
<td>get new model node instance index</td>
</tr>
<tr>
<td><strong>ResetModelNodeInstanceIndex ()</strong></td>
<td>reset frame model node instance index</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>int</td>
<td><code>GetRefCount()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>AddRef()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>Release()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsInstanceOf(const Util::String &amp;className)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA(const Rtti &amp;rtti)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA(const Util::String &amp;rttiName)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
</tr>
<tr>
<td>const</td>
<td><code>GetClassName()</code></td>
</tr>
<tr>
<td><code>Util::FourCC</code></td>
<td><code>GetClassFourCC()</code></td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

const Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
This method should be called as the very last before an application exits.
Models::ShapeNode
Models::ShapeNode Class Reference

#include <shapenode.h>

Inheritance diagram for Models::ShapeNode:
Detailed Description

**ModelNode** which describes a simple static shape. Since the **ShapeNode** is derived from the **StateNode** (which in turn is derived from the **TransformNode**), a **ShapeNode** contains all the necessary information for rendering a mesh.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShapeNode ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~ShapeNode ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual <strong>Ptr</strong>&lt; <strong>ModelNodeInstance</strong> &gt; CreateNodeInstance () const</td>
<td>create a model node instance</td>
</tr>
<tr>
<td>virtual bool ParseDataTag (const <strong>Util::FourCC</strong> &amp;fourCC, const <strong>Ptr</strong>&lt; <strong>IO::BinaryReader</strong> &gt; &amp;reader)</td>
<td>parse data tag (called by loader code)</td>
</tr>
<tr>
<td>virtual void LoadResources ()</td>
<td>called when resources should be loaded</td>
</tr>
<tr>
<td>virtual void UnloadResources ()</td>
<td>called when resources should be unloaded</td>
</tr>
<tr>
<td>virtual <strong>Resources::Resource::State</strong> GetResourceState () const</td>
<td>get overall state of contained resources (Init, Loaded, Pending, Failed, Cancelled)</td>
</tr>
<tr>
<td>virtual void ApplySharedState (IndexT frameIndex)</td>
<td>apply state shared by all my ModelNodeInstances</td>
</tr>
<tr>
<td>void SetMeshResourceId (const <strong>Resources::Resource::ResourceId</strong> &amp;resId)</td>
<td>set mesh resource id</td>
</tr>
<tr>
<td>const <strong>Resources::Resource::ResourceId</strong> &amp; GetMeshResourceId () const</td>
<td>get mesh resource id</td>
</tr>
<tr>
<td>void SetPrimitiveGroupIndex (IndexT)</td>
<td>set primitive group index</td>
</tr>
<tr>
<td>IndexT GetPrimitiveGroupIndex () const</td>
<td>get primitive group index</td>
</tr>
<tr>
<td>void SetMeshResourceLoader (const <strong>Resources::ResourceLoader</strong> &amp;loader)</td>
<td>set optional resourceloader</td>
</tr>
<tr>
<td>const <strong>Ptr</strong>&lt; <strong>Resources::ManagedMesh</strong> &gt; &amp; GetManagedMesh () const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>SetManagedMesh</code></td>
<td>Get managed mesh</td>
</tr>
<tr>
<td><code>void SetShader (const Resources::ResourceId &amp;resId)</code></td>
<td>Set shader resource id</td>
</tr>
<tr>
<td><code>const Resources::ResourceId &amp; GetShader () const</code></td>
<td>Get shader resource id</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp; GetShaderInstance () const</code></td>
<td>Get pointer to contained shader instance</td>
</tr>
<tr>
<td><code>void AddShaderParam (const Util::String &amp;paramName, const Util::Variant &amp;paramValue)</code></td>
<td>Add optional shader parameter, must be called before LoadResources</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Util::KeyValuePair&lt; Util::StringAtom, Util::Variant &gt; &gt; &amp; GetShaderParameter () const</code></td>
<td>Get shaderparams</td>
</tr>
<tr>
<td><code>void SetPosition (const Math::point &amp;p)</code></td>
<td>Set position</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetPosition () const</code></td>
<td>Get position</td>
</tr>
<tr>
<td><code>void SetRotation (const Math::quaternion &amp;r)</code></td>
<td>Set rotate quaternion</td>
</tr>
<tr>
<td><code>const Math::quaternion &amp; GetRotation () const</code></td>
<td>Get rotate quaternion</td>
</tr>
<tr>
<td><code>void SetScale (const Math::vector &amp;s)</code></td>
<td>Set scale</td>
</tr>
<tr>
<td><code>const Math::vector &amp; GetScale () const</code></td>
<td>Get scale</td>
</tr>
<tr>
<td><code>void SetRotatePivot (const Math::point &amp;p)</code></td>
<td>Set rotate pivot</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetRotatePivot () const</code></td>
<td>Get rotate pivot</td>
</tr>
<tr>
<td><code>void SetScalePivot (const Math::point &amp;p)</code></td>
<td>Set scale pivot</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetScalePivot () const</code></td>
<td>Get scale pivot</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool <code>IsInViewSpace</code> (const)</td>
<td>is transformnode in viewspace</td>
</tr>
<tr>
<td>void <code>SetInViewSpace</code> (bool b)</td>
<td>set transformnode in viewspace</td>
</tr>
<tr>
<td>float <code>GetMinDistance</code> (const)</td>
<td>get MinDistance</td>
</tr>
<tr>
<td>void <code>SetMinDistance</code> (float val)</td>
<td>set MinDistance</td>
</tr>
<tr>
<td>float <code>GetMaxDistance</code> (const)</td>
<td>get MaxDistance</td>
</tr>
<tr>
<td>void <code>SetMaxDistance</code> (float val)</td>
<td>set MaxDistance</td>
</tr>
<tr>
<td>bool <code>LodDistancesUsed</code> (const)</td>
<td>are lod distances used</td>
</tr>
<tr>
<td>virtual void <code>OnAttachToModel</code> (const <code>Ptr&lt;Model&gt;</code> &amp;model)</td>
<td>called when attached to model node</td>
</tr>
<tr>
<td>bool <code>GetLockedToViewer</code> (const)</td>
<td>get LockedToViewer</td>
</tr>
<tr>
<td>void <code>SetLockedToViewer</code> (bool val)</td>
<td>set LockedToViewer</td>
</tr>
<tr>
<td>bool <code>CheckLodDistance</code> (float distToViewer)</td>
<td>helper method to check whether the distance is within lod distances</td>
</tr>
<tr>
<td><code>Ptr&lt;ModelNodeInstance&gt;</code> <code>CreateNodeInstanceHierarchy</code> (const <code>Ptr&lt;Model&gt;</code> &amp;model)</td>
<td>recursively create model node instance and model node instances</td>
</tr>
<tr>
<td>virtual void <code>OnRemoveFromModel</code> ()</td>
<td>called when removed from model node</td>
</tr>
<tr>
<td>virtual void <code>OnResourcesLoaded</code> ()</td>
<td>called once when all pending resource have been loaded</td>
</tr>
<tr>
<td>virtual void <code>BeginParseDataTags</code> ()</td>
<td>begin parsing data tags</td>
</tr>
<tr>
<td>virtual void <code>EndParseDataTags</code> ()</td>
<td>finish parsing data tags</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsAttachedToModel()</code></td>
<td>const return true if currently attached to a Model</td>
</tr>
<tr>
<td><code>const Ptr&lt; Model &gt; &amp; GetModel()</code></td>
<td>get model this node is attached to</td>
</tr>
<tr>
<td><code>void SetResourceStreamingLevelOfDetail(float factor)</code></td>
<td>sets the resourceStreamingLevelOfDetail but only if the given value is bigger than the current one (reset on frame-start)</td>
</tr>
<tr>
<td><code>void ResetScreenSpaceStats()</code></td>
<td>resets all screen space stats e.g. size</td>
</tr>
<tr>
<td><code>void SetBoundingBox(const Math::bbox &amp;b)</code></td>
<td>set bounding box</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetBoundingBox()</code></td>
<td>get bounding box of model node</td>
</tr>
<tr>
<td><code>void SetName(const Util::StringAtom </code></td>
<td>set model node name</td>
</tr>
<tr>
<td><code>const Util::StringAtom &amp; GetName()</code></td>
<td>get model node name</td>
</tr>
<tr>
<td><code>void SetType(ModelNodeType::Code ModelNodeType::Code)</code></td>
<td>set ModelNodeType</td>
</tr>
<tr>
<td><code>getModelType()</code></td>
<td>get the ModelNodeType</td>
</tr>
<tr>
<td><code>void SetParent(const Ptr&lt; ModelNode &gt; &amp;p)</code></td>
<td>set parent node</td>
</tr>
<tr>
<td><code>const Ptr&lt; ModelNode &gt; &amp; GetParent()</code></td>
<td>get parent node</td>
</tr>
<tr>
<td><code>bool HasParent()</code></td>
<td>const return true if node has a parent</td>
</tr>
<tr>
<td><code>void AddChild(const Ptr&lt; ModelNode &gt; &amp;c)</code></td>
<td>add a child node</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Ptr&lt; ModelNode &gt; &gt; &amp; GetChildren()</code></td>
<td>get child nodes</td>
</tr>
<tr>
<td><code>bool HasChild(const Util::StringAtom)</code></td>
<td>get child nodes</td>
</tr>
</tbody>
</table>
const &name) const

return true if a direct child exists by name

const Ptr< ModelNode > &

LookupChild (const Util::StringAtom &name) const

get pointer to direct child by name

void

AddVisibleNodeInstance (IndexT frameIndex, const &nodeInst)
called by model node instance on NotifyVisible()

const Util::Array< &nodeInst

GetVisibleModelNodeInstances (ModelNodeType::Code t) const
get visible model node instances

bool

HasStringAttr (const &attrId) const

has string attr

const &

GetStringAttr (const &attrId) const

get string attr

void

SetStringAttr (const &attrId, const &value)

add string attribute

int

GetRefCount () const

get the current refcount

void

AddRef ()
iccrement refcount by one

void

Release ()
decrement refcount and destroy object if refcount is zero

bool

IsInstanceOf (const Rtti &rtti) const

return true if this object is instance of given class

bool

IsInstanceOf (const Util::String &className) const

return true if this object is instance of given class by string

bool

IsInstanceOf (const Util::FourCC &classFourCC) const

return true if this object is instance of given class
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetupManagedTextureVariable (const Resources::ResourceId &amp;texResId, const Ptr<a href="">CoreGraphics::ShaderVariable</a> &amp;var)</td>
<td>setup a new managed texture variable</td>
</tr>
<tr>
<td>UpdateManagedTextureVariables (IndexT frameIndex)</td>
<td>update managed texture variables</td>
</tr>
<tr>
<td>RecurseCreateNodeInstanceHierarchy (const Ptr&lt;ModelInstance&gt; &amp;modelInst, const Ptr&lt;ModelNodeInstance&gt; &amp;parentNodeInst)</td>
<td>recursively create node instance hierarchy</td>
</tr>
</tbody>
</table>
### Protected Attributes

<table>
<thead>
<tr>
<th>float</th>
<th><code>resourceStreamingLevelOfDetail</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>factor between 0.0 (close) and 1.0 (far away) describing the distance to camera (used for decision of max needed mipMap)</em></td>
</tr>
</tbody>
</table>
add optional shader parameter, must be called before LoadResources

Manual shader parameters must be added before LoadResources is called, because on LoadResources all shaderparams are validated.

setup a new managed texture variable

Create a new managed texture resource and bind it to the provided shader variable.

update managed texture variables

This method transfers texture from our managed texture objects into their associated shader variable. This is necessary since the actual texture of a managed texture may change from frame to frame because of resource management.
recursively create model node instance and child model node instances

Create the node instance hierarchy.

```cpp
class ModelNode
{
public:
    virtual void OnResourcesLoaded() = 0;

    virtual void BeginParseDataTags() = 0;

    virtual void EndParseDataTags() = 0;

    void ResetScreenSpaceStats()
    {
        // Reset resourceStreamingLevelOfDetail to -1.0 as we are able
to recognize invisible items this way. (visible items will overwrite this
value with a value >= 0.0)
    }
};
```

called once when all pending resource have been loaded

This method is called once by Model::OnResourcesLoaded() when all pending resources of a model have been loaded.

Reimplemented in `Characters::CharacterNode`, and `Particles::ParticleSystemNode`.

```
void Models::ModelNode::OnResourcesLoaded()
{
    // called once when all pending resource have been loaded
}
```

```
void Models::ModelNode::BeginParseDataTags()
{
    // begin parsing data tags
}
```

```
void Models::ModelNode::EndParseDataTags()
{
    // finish parsing data tags
}
```

```
void Models::ModelNode::ResetScreenSpaceStats()
{
    // Reset resourceStreamingLevelOfDetail to -1.0 as we are able to
recognize invisible items this way. (visible items will overwrite this
value with a value >= 0.0)
}
```
Ptr< ModelNodeInstance >
Models::ModelNode::RecurseCreateNodeInstanceHierarchy ( const Ptr< ModelInstance > & modelInst, const Ptr< ModelNodeInstance > parentNodeInst )

recursively create node instance hierarchy

Recursively create node instances and attach them to the provided model instance. Returns a pointer to the root node instance.

int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Models::ShapeNodeInstance
Models::ShapeNodeInstance Class Reference

#include <shapenodeinstance.h>

Inheritance diagram for Models::ShapeNodeInstance:
Detailed Description

The ShapeNodeInstance actually renders a static shape, and holds all the necessary per-instance state to do so.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShapeNodeInstance ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~ShapeNodeInstance ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>OnVisibilityResolve</strong> (IndexT resolveIndex, float distToViewer)**</td>
<td>Called during visibility resolve</td>
</tr>
<tr>
<td>virtual void <strong>Render ()</strong></td>
<td>Perform rendering</td>
</tr>
<tr>
<td>virtual void <strong>ApplyState ()</strong></td>
<td>Apply per-instance state prior to rendering</td>
</tr>
<tr>
<td><strong>Ptr&lt; CoreGraphics::ShaderVariableInstance &gt;</strong> CreateShaderVariableInstance** (CoreGraphics::ShaderVariable::Semantic &amp;semantic)**</td>
<td>Instantiate a shader variable by semantic</td>
</tr>
<tr>
<td>bool <strong>HasShaderVariableInstance</strong> (CoreGraphics::ShaderVariable::Semantic &amp;semantic) const **</td>
<td>Return true if a shader variable has been instantiated</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; CoreGraphics::ShaderVariableInstance &gt;</strong> &amp; <strong>GetShaderVariableInstance</strong> (CoreGraphics::ShaderVariable::Semantic &amp;semantic) const **</td>
<td>Get a shader variable instance</td>
</tr>
<tr>
<td>virtual void <strong>OnRenderBefore</strong> (IndexT frameIndex, Timing::Time time)**</td>
<td>Called from ModelEntity::OnRenderBefore</td>
</tr>
<tr>
<td>void <strong>GetPosition ()</strong> const <strong>GetPosition</strong> (const Math::point &amp;)**</td>
<td>Set position</td>
</tr>
<tr>
<td>const Math::point &amp; <strong>GetPosition</strong> () const <strong>GetPosition</strong> (const Math::point &amp;)**</td>
<td>Get position</td>
</tr>
<tr>
<td>void <strong>SetRotate</strong> (const Math::quaternion &amp;)**</td>
<td>Set rotate quaternion</td>
</tr>
<tr>
<td>const Math::quaternion &amp; <strong>GetRotate</strong> () const <strong>GetRotate</strong> (const Math::quaternion &amp;) **</td>
<td>Get rotate quaternion</td>
</tr>
</tbody>
</table>
void SetScale (const Math::vector & set scale)

const Math::vector & GetScale () const
get scale

void SetRotatePivot (const Math::point & set rotate pivot)

const Math::point & GetRotatePivot () const
get rotate pivot

void SetScalePivot (const Math::point & set scale pivot)

const Math::point & GetScalePivot () const
get scale pivot

void SetOffsetMatrix (const Math::matrix44 &m)
set optional offset matrix

const Math::matrix44 & GetOffsetMatrix () const
get optional offset matrix

bool IsInViewSpace () const
is transformnode in viewspace

void SetInViewSpace (bool b)
set transformnode in viewspace

bool GetLockedToViewer () const
get LockedToViewer

void SetLockedToViewer (bool val)
set LockedToViewer

const Math::matrix44 & GetLocalTransform ()
get resulting local transform matrix in local parent space

const Math::matrix44 & GetModelTransform ()
get model space transform (valid after Update())

virtual void DiscardHierarchy ()
discard the model node instance and all of its children

bool IsValid () const
return true if the model node instance is valid

virtual void OnNotifyCullingVisible (IndexT frameIndex, Timing::Time time)
called from ModelEntity::OnNotifyCullingVisible
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetName () const</td>
<td>get model node name</td>
</tr>
<tr>
<td>HasParent () const</td>
<td>return true if node has a parent</td>
</tr>
<tr>
<td>GetParent () const</td>
<td>get parent node</td>
</tr>
<tr>
<td>GetChildren () const</td>
<td>get child nodes</td>
</tr>
<tr>
<td>HasChild (const Util::StringAtom &amp;name) const</td>
<td>return true if a direct child exists by name</td>
</tr>
<tr>
<td>LookupChild (const Util::StringAtom &amp;name) const</td>
<td>get pointer to direct child by name</td>
</tr>
<tr>
<td>LookupPath (const Util::String &amp;hierarchy) const</td>
<td>get modelnodeinstance by hierarchy path</td>
</tr>
<tr>
<td>GetModellInstance () const</td>
<td>get the ModellInstance we are attached</td>
</tr>
<tr>
<td>GetModelNode () const</td>
<td>get the ModelNode we’re associated</td>
</tr>
<tr>
<td>SetVisible (bool b, Timing::Time recursive=true)</td>
<td>set the node instance’s visibility</td>
</tr>
<tr>
<td>IsVisible () const</td>
<td>return true if node instance is set to visible</td>
</tr>
<tr>
<td>GetModelNodeInstanceIndex ()</td>
<td>get model node instance index for current frame</td>
</tr>
<tr>
<td>GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>Release ()</td>
<td>decrement refcount and destroy object</td>
</tr>
<tr>
<td>IsInstanceOf (const Rtti &amp;rtti)</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class or derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class or derived class by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class or derived class by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Discard()</code></td>
<td>called when removed from <code>ModelInstance</code></td>
</tr>
<tr>
<td><code>Setup(const Ptr&lt; ModelInstance &gt;&amp; inst, const Ptr&lt; ModelNode &gt;&amp; node, const Ptr&lt; ModelNodeInstance &gt;&amp; parentNodeInst)</code></td>
<td>called when attached to <code>ModelInstance</code></td>
</tr>
<tr>
<td><code>RenderDebug()</code></td>
<td>render node specific debug shape</td>
</tr>
<tr>
<td><code>SetParent(const Ptr&lt; ModelNodeInstance &gt;&amp; p)</code></td>
<td>set parent node</td>
</tr>
<tr>
<td><code>AddChild(const Ptr&lt; ModelNodeInstance &gt;&amp; c)</code></td>
<td>add a child node</td>
</tr>
<tr>
<td><code>OnShow(Timing::Time time)</code></td>
<td>called when the node becomes visible with current time</td>
</tr>
<tr>
<td><code>OnHide(Timing::Time time)</code></td>
<td>called when the node becomes invisible</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Models::TransformNodeInstance::OnRenderBefore ( IndexT frameIndex, Timing::Time time ) [virtual, inherited]
called from ModelEntity::OnRenderBefore

The update method should first invoke any animators which change per-instance attributes (this is done in the parent class). Then the local space transforms must be flattened into model space.

NOTE: this method must be called late in the frame to give other systems a chance to modify the transform matrix (for instance the character attachment system).

Reimplemented from Models::ModelNodeInstance.
Reimplemented in Particles::ParticleSystemNodeInstance.

void Models::TransformNodeInstance::RenderDebug ( ) [protected, virtual, inherited]
render node specific debug shape
Render a debug visualization of the node.
Reimplemented from Models::ModelNodeInstance.
Reimplemented in Characters::CharacterNodeInstance, and Particles::ParticleSystemNodeInstance.

void Models::ModelNodeInstance::DiscardHierarchy ( ) [virtual, inherited]
discard the model node instance and all of its children
Discards this model node instance and all of its children recursively.
bool Models::ModelNodeInstance::isVisible() const [inherited]

return true if node instance is set to visible.

FIXME: The recursion in this is method makes it slow, especially in deep hierarchies. Also it can't be inlined.

int Core::RefCounted::getRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks()
[static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Models::StateNode
Models::StateNode Class Reference

#include <statenode.h>

Inheritance diagram for Models::StateNode:
Detailed Description

A model node which holds shader state information and applies shared shader state for its set of StateNodeInstances.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StateNode ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~StateNode ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual <strong>Ptr&lt; ModelNodeInstance &gt;</strong></td>
<td>CreateNodeInstance () const create a model node instance</td>
</tr>
<tr>
<td>virtual <strong>bool</strong> ParseDataTag (const Util::FourCC &amp;fourCC, const <strong>Ptr&lt; IO::BinaryReader &gt;</strong> &amp;reader)</td>
<td>parse data tag (called by loader code)</td>
</tr>
<tr>
<td>virtual void <strong>LoadResources ()</strong></td>
<td>called when resources should be loaded</td>
</tr>
<tr>
<td>virtual void <strong>UnloadResources ()</strong></td>
<td>called when resources should be unloaded</td>
</tr>
<tr>
<td>virtual <strong>Resources::Resource::State</strong></td>
<td>GetResourceState () const get overall state of contained resources (Initial, Loaded, Pending, Failed, Cancelled)</td>
</tr>
<tr>
<td>virtual void <strong>ApplySharedState</strong> (IndexT frameIndex)</td>
<td>apply state shared by all my ModelNodeInstances</td>
</tr>
<tr>
<td>void <strong>SetShader</strong> (const <strong>Resources::ResourceId</strong> &amp;resId)</td>
<td>set shader resource id</td>
</tr>
<tr>
<td>const <strong>Resources::ResourceId</strong> &amp; <strong>GetShader ()</strong> const</td>
<td>get shader resource id</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; CoreGraphics::ShaderInstance &gt;</strong> &amp; <strong>GetShaderInstance ()</strong> const</td>
<td>get pointer to contained shader instance</td>
</tr>
<tr>
<td>void <strong>AddShaderParam</strong> (const Util::String &amp;paramName, const Util::Variant &amp;paramValue)</td>
<td>add optional shader parameter, must be called before LoadResources</td>
</tr>
<tr>
<td>const <strong>Util::Array&lt; Util::KeyValuePair&lt; Util::StringAtom, Util::Variant &gt; &gt;</strong> &amp; <strong>GetShaderParameter ()</strong> const</td>
<td>get shaderparams</td>
</tr>
</tbody>
</table>
void SetPosition (const Math::point &p)  
set position

const Math::point & GetPosition () const  
get position

void SetRotation (const Math::quaternion &r)  
set rotate quaternion

const Math::quaternion & GetRotation () const  
get rotate quaternion

void SetScale (const Math::vector &s)  
set scale

const Math::vector & GetScale () const  
get scale

void SetRotatePivot (const Math::point &p)  
set rotate pivot

const Math::point & GetRotatePivot () const  
get rotate pivot

void SetScalePivot (const Math::point &p)  
set scale pivot

const Math::point & GetScalePivot () const  
get scale pivot

bool IsInViewSpace () const  
is transformnode in viewspace

void SetInViewSpace (bool b)  
set transformnode in viewspace

float GetMinDistance () const  
get MinDistance

void SetMinDistance (float val)  
set MinDistance

float GetMaxDistance () const  
get MaxDistance

void SetMaxDistance (float val)  
set MaxDistance

bool LodDistancesUsed () const  
are lod distances used
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void OnAttachToModel (const Ptr&lt; Model&gt; &amp;model)</td>
<td>called when attached to model node</td>
</tr>
<tr>
<td>bool GetLockedToViewer () const</td>
<td>get LockedToViewer</td>
</tr>
<tr>
<td>void SetLockedToViewer (bool val)</td>
<td>set LockedToViewer</td>
</tr>
<tr>
<td>bool CheckLodDistance (float distToViewer) const</td>
<td>helper method to check whether the distance within lod distances</td>
</tr>
<tr>
<td>Ptr&lt; ModelNodeInstance &gt; CreateNodeInstanceHierarchy</td>
<td>recursively create model node instance and model node instances</td>
</tr>
<tr>
<td>virtual void OnRemoveFromModel ()</td>
<td>called when removed from model node</td>
</tr>
<tr>
<td>virtual void OnResourcesLoaded ()</td>
<td>called once when all pending resource have loaded</td>
</tr>
<tr>
<td>virtual void BeginParseDataTags ()</td>
<td>begin parsing data tags</td>
</tr>
<tr>
<td>virtual void EndParseDataTags ()</td>
<td>finish parsing data tags</td>
</tr>
<tr>
<td>bool IsAttachedToModel () const</td>
<td>return true if currently attached to a Model</td>
</tr>
<tr>
<td>const Ptr&lt; Model &gt; &amp; GetModel () const</td>
<td>get model this node is attached to</td>
</tr>
<tr>
<td>void SetResourceStreamingLevelOfDetail (float factor)</td>
<td>sets the resourceStreamingLevelOfDetail but the given value is bigger than the current one (reseted on frame-start)</td>
</tr>
<tr>
<td>void ResetScreenSpaceStats ()</td>
<td>resets all screen space stats e.g. size</td>
</tr>
<tr>
<td>void SetBoundingBox (const Math::bbox &amp;b)</td>
<td>set bounding box</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetBoundingBox()</code></td>
<td>Get bounding box of model node</td>
</tr>
<tr>
<td><code>SetName()</code></td>
<td>Set model node name</td>
</tr>
<tr>
<td><code>GetName()</code></td>
<td>Get model node name</td>
</tr>
<tr>
<td><code>setType()</code></td>
<td>Set <code>ModelNodeType::Code</code></td>
</tr>
<tr>
<td><code>SetParent()</code></td>
<td>Set parent node</td>
</tr>
<tr>
<td><code>GetParent()</code></td>
<td>Get parent node</td>
</tr>
<tr>
<td><code>HasParent()</code></td>
<td>Return true if node has a parent</td>
</tr>
<tr>
<td><code>AddChild()</code></td>
<td>Add a child node</td>
</tr>
<tr>
<td><code>GetChildren()</code></td>
<td>Get child nodes</td>
</tr>
<tr>
<td><code>HasChild()</code></td>
<td>Return true if a direct child exists by name</td>
</tr>
<tr>
<td><code>LookupChild()</code></td>
<td>Get pointer to direct child by name</td>
</tr>
<tr>
<td><code>AddVisibleNodeInstance()</code></td>
<td>Called by model node instance on NotifyVisible()</td>
</tr>
<tr>
<td><code>GetVisibleModelNodeInstances()</code></td>
<td>Get visible model node instances</td>
</tr>
<tr>
<td><code>HasStringAttr()</code></td>
<td>Return true if a StringAttribute exists by name</td>
</tr>
</tbody>
</table>
has string attr

const Util::StringAtom &

**GetStringAttr** (const Util::StringAtom &attrId) const

get string attr

void

**SetStringAttr** (const Util::StringAtom &attrId, const Util::StringAtom &value)

add string attribute

int

**GetRefCount** () const

get the current refcount

void

**AddRef** ()

increment refcount by one

void

**Release** ()

decrement refcount and destroy object if refcount is zero

bool

**IsInstanceOf** (const Rtti &rtti) const

return true if this object is instance of given class

bool

**IsInstanceOf** (const Util::String &className) const

return true if this object is instance of given class by string

bool

**IsInstanceOf** (const Util::FourCC &classFourCC) const

return true if this object is instance of given class by fourcc

bool

**IsA** (const Rtti &rtti) const

return true if this object is instance of given class, or a derived class

bool

**IsA** (const Util::String &rttiName) const

return true if this object is instance of given class, by string

bool

**IsA** (const Util::FourCC &rttiFourCC) const

return true if this object is instance of given class, by fourcc

const Util::String &

**GetClassName** () const

get the class name

Util::FourCC

**GetClassFourCC** () const
get the class FourCC code
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetupManagedTextureVariable (const Resources::ResourceId &amp;texResId, const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp;var)</code></td>
<td>setup a new managed texture variable</td>
</tr>
<tr>
<td><code>void UpdateManagedTextureVariables (IndexT frameIndex)</code></td>
<td>update managed texture variables</td>
</tr>
<tr>
<td><code>virtual Ptr&lt; ModelNodeInstance &gt; RecurseCreateNodeInstanceHierarchy (const Ptr&lt; ModelInstance &gt; &amp;modelInst, const Ptr&lt; ModelNodeInstance &gt; &amp;parentNodeInst)</code></td>
<td>recursively create node instance hierarchy</td>
</tr>
</tbody>
</table>
### Protected Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>resourceStreamingLevelOfDetail</code></td>
<td>Factor between 0.0 (close) and 1.0 (far away) describing the distance to camera (used for decision of max needed mipMap)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::StateNode::AddShaderParam
    ( const Util::String & paramName,
      const Util::Variant & paramValue
    )
```

add optional shader parameter, must be called before LoadResources

Manual shader parameters must be added before LoadResources is called, because on LoadResources all shaderparams are validated.

```cpp
void Models::StateNode::SetupManagedTextureVariable
    ( const Resources::ResourceId & texResId,
      const Ptr< CoreGraphics::ShaderVariable > & var
    ) [protected]
```

setup a new managed texture variable

Create a new managed texture resource and bind it to the provided shader variable.

```cpp
void Models::StateNode::UpdateManagedTextureVariables
    ( IndexT frameIndex ) [protected]
```

update managed texture variables

This method transfers texture from our managed texture objects into their associated shader variable. This is necessary since the actual texture of a managed texture may change from frame to frame because of resource management.
recursively create model node instance and child model node instances

Create the node instance hierarchy.

```cpp
void Models::ModelNode::OnResourcesLoaded( ) [virtual, inherited]
called once when all pending resource have been loaded
This method is called once by Model::OnResourcesLoaded() when all pending resources of a model have been loaded.
Reimplemented in Characters::CharacterNode, and Particles::ParticleSystemNode.
```

```cpp
void Models::ModelNode::BeginParseDataTags( ) [virtual, inherited]
begin parsing data tags
Begin parsing data tags. This method is called by StreamModelLoader before ParseDataTag() is called for the first time.
```

```cpp
void Models::ModelNode::EndParseDataTags( ) [virtual, inherited]
finish parsing data tags
End parsing data tags. This method is called by StreamModelLoader after the last ParseDataTag() is called.
```

```cpp
void Models::ModelNode::ResetScreenSpaceStats( ) [inline, inherited]
resets all screen space stats e.g. size
Reset resourceStreamingLevelOfDetail to -1.0 as we are able to recognize invisible items this way. (visible items will overwrite this value with a value >= 0.0)```
Ptr< ModelNodeInstance >
Models::ModelNode::RecurseCreateNodeInstanceHierarchy(
    const Ptr< ModelInstance >& modelInst,
    const Ptr< ModelNodeInstance >& parentNodeInst
)

recursively create node instance hierarchy

Recursively create node instances and attach them to the provided model instance. Returns a pointer to the root node instance.

int
Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC


get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Models::StateNodeInstance
Models::StateNodeInstance Class Reference

#include <statenodeinstance.h>

Inheritance diagram for Models::StateNodeInstance:

- Core::RefCounted
- Models::ModelNodeInstance
- Models::TransformNodeInstance
- Models::StateNodeInstance
- Models::ShapeNodeInstance
- Particles::ParticleSystemNodeInstance
- Characters::CharacterSkinNodeInstance
Detailed Description

Holds and applies per-instance shader state.

(C) 2007 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StateNodeInstance ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~StateNodeInstance ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <code>ApplyState ()</code></td>
<td>apply per-instance state prior to rendering</td>
</tr>
<tr>
<td><code>Ptr&lt; CoreGraphics::ShaderVariableInstance &gt; CreateShaderVariableInstance (CoreGraphics::ShaderVariable::Semantic &amp;semantic)</code></td>
<td>instanciate a shader variable by semantic</td>
</tr>
<tr>
<td>bool <code>HasShaderVariableInstance (CoreGraphics::ShaderVariable::Semantic &amp;semantic) const</code></td>
<td>return true if a shader variable has been instanciated</td>
</tr>
<tr>
<td>const <code>Ptr&lt; CoreGraphics::ShaderVariableInstance &gt; &amp; GetShaderVariableInstance (CoreGraphics::ShaderVariable::Semantic &amp;semantic) const</code></td>
<td>get a shader variable instance</td>
</tr>
<tr>
<td>virtual void <code>OnRenderBefore (IndexT frameIndex, Timing::Time time)</code></td>
<td>called from ModelEntity::OnRenderBefore</td>
</tr>
<tr>
<td>void <code>GetPosition ()</code> const</td>
<td>get position</td>
</tr>
<tr>
<td>const <code>Math::point &amp; GetPosition ()</code> const</td>
<td>get position</td>
</tr>
<tr>
<td>void <code>SetPosition (const Math::point &amp;)</code></td>
<td>set position</td>
</tr>
<tr>
<td>void <code>SetRotate (const Math::quat &amp;)</code></td>
<td>set rotate quaternion</td>
</tr>
<tr>
<td>const <code>Math::quat &amp; GetRotate ()</code> const</td>
<td>get rotate quaternion</td>
</tr>
<tr>
<td>void <code>SetScale (const Math::vec3 &amp;)</code></td>
<td>set scale</td>
</tr>
<tr>
<td>const <code>Math::vec3 &amp; GetScale ()</code> const</td>
<td>get scale</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>SetRotatePivot</code> (const <code>Math::point</code> &amp;)</td>
<td>set rotate pivot</td>
</tr>
<tr>
<td><code>GetRotatePivot</code> () const</td>
<td>get rotate pivot</td>
</tr>
<tr>
<td><code>SetScalePivot</code> (const <code>Math::point</code> &amp;)</td>
<td>set scale pivot</td>
</tr>
<tr>
<td><code>GetScalePivot</code> () const</td>
<td>get scale pivot</td>
</tr>
<tr>
<td><code>SetOffsetMatrix</code> (const <code>Math::matrix44</code> &amp;m)</td>
<td>set optional offset matrix</td>
</tr>
<tr>
<td><code>GetOffsetMatrix</code> () const</td>
<td>get optional offset matrix</td>
</tr>
<tr>
<td><code>IsInViewSpace</code> () const</td>
<td>is transform node in viewspace</td>
</tr>
<tr>
<td><code>SetInViewSpace</code> (bool b)</td>
<td>set transform node in viewspace</td>
</tr>
<tr>
<td><code>GetLockedToViewer</code> () const</td>
<td>get LockedToViewer</td>
</tr>
<tr>
<td><code>SetLockedToViewer</code> (bool val)</td>
<td>set LockedToViewer</td>
</tr>
<tr>
<td><code>GetLocalTransform</code> ()</td>
<td>get resulting local transform matrix in local parent space</td>
</tr>
<tr>
<td><code>GetModelTransform</code> () const</td>
<td>get model space transform (valid after Update())</td>
</tr>
<tr>
<td><code>DiscardHierarchy</code> ()</td>
<td>discard the model node instance and all of its children</td>
</tr>
<tr>
<td><code>IsValid</code> () const</td>
<td>return true if the model node instance is valid</td>
</tr>
<tr>
<td><code>OnNotifyCullingVisible</code> (IndexT frameIndex, <code>Timing::Time</code> time)</td>
<td>called from <code>ModelEntity::OnNotifyCullingVisible</code></td>
</tr>
<tr>
<td><code>OnVisibilityResolve</code> (IndexT resolveIndex, float distanceToViewer)</td>
<td>called during visibility resolve</td>
</tr>
<tr>
<td><code>Render</code> ()</td>
<td>perform rendering</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetName()</code></td>
<td>Get model node name</td>
</tr>
<tr>
<td><code>HasParent()</code></td>
<td>Return true if node has a parent</td>
</tr>
<tr>
<td><code>GetParent()</code></td>
<td>Get parent node</td>
</tr>
<tr>
<td><code>GetChildren()</code></td>
<td>Get child nodes</td>
</tr>
<tr>
<td><code>HasChild(const Util::StringAtom &amp;)</code></td>
<td>Return true if a direct child exists by name</td>
</tr>
<tr>
<td><code>LookupChild(const Util::StringAtom &amp;name)</code></td>
<td>Get pointer to direct child by name</td>
</tr>
<tr>
<td><code>LookupPath(const Util::String &amp;)</code></td>
<td>Get modelnodeinstance by hierarchy path</td>
</tr>
<tr>
<td><code>GetModellInstance()</code></td>
<td>Get the <code>ModellInstance</code> we are attached</td>
</tr>
<tr>
<td><code>GetModelNode()</code></td>
<td>Get the <code>ModelNode</code> we're associated with</td>
</tr>
<tr>
<td><code>SetVisible(bool b, Timing::Time recursive=true)</code></td>
<td>Set the node instance's visibility</td>
</tr>
<tr>
<td><code>IsVisible()</code></td>
<td>Return true if node instance is set to visible</td>
</tr>
<tr>
<td><code>GetModelNode_InstanceIndex()</code></td>
<td>Get model node instance index for current frame</td>
</tr>
<tr>
<td><code>GetRefCount()</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;)</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC&amp; classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA (const Rtti&amp; rtti)</code></td>
<td>return true if this object is instance of given class or a derived class</td>
</tr>
<tr>
<td><code>IsA (const Util::String&amp; rttiName)</code></td>
<td>return true if this object is instance of given class or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC&amp; rttiFourCC)</code></td>
<td>return true if this object is instance of given class or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void Discard ()</td>
<td><em>called when removed from ModelInstance</em></td>
</tr>
<tr>
<td>virtual void Setup (const Ptr&lt;ModelInstance&gt; &amp;inst, const Ptr&lt;ModelNode&gt; &amp;node, const Ptr&lt;ModelNodeInstance&gt; &amp;parentNodeInst)</td>
<td><em>called when attached to ModelInstance</em></td>
</tr>
<tr>
<td>virtual void RenderDebug ()</td>
<td><em>render node specific debug shape</em></td>
</tr>
<tr>
<td>void SetParent (const Ptr&lt;ModelNodeInstance&gt; &amp;p)</td>
<td><em>set parent node</em></td>
</tr>
<tr>
<td>void AddChild (const Ptr&lt;ModelNodeInstance&gt; &amp;c)</td>
<td><em>add a child node</em></td>
</tr>
<tr>
<td>virtual void OnShow (Timing::Time time)</td>
<td><em>called when the node becomes visible with current time</em></td>
</tr>
<tr>
<td>virtual void OnHide (Timing::Time time)</td>
<td><em>called when the node becomes invisible</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::TransformNodeInstance::OnRenderBefore ( IndexT frameIndex,
                                                      Timing::Time time ) [virtual, inherited]
```
called from ModelEntity::OnRenderBefore

The update method should first invoke any animators which change per-instance attributes (this is done in the parent class). Then the local space transforms must be flattened into model space.

NOTE: this method must be called late in the frame to give other systems a chance to modify the transform matrix (for instance the character attachment system).

Reimplemented from `Models::ModelNodeInstance`.
Reimplemented in `Particles::ParticleSystemNodeInstance`.

```cpp
void Models::TransformNodeInstance::RenderDebug ( ) [protected, virtual, inherited]
```
render node specific debug shape

Render a debug visualization of the node.

Reimplemented from `Models::ModelNodeInstance`.
Reimplemented in `Characters::CharacterNodeInstance`, and `Particles::ParticleSystemNodeInstance`.

```cpp
void Models::ModelNodeInstance::DiscardHierarchy ( ) [virtual, inherited]
```
discard the model node instance and all of its children

Discards this model node instance and all of its children recursively.
void Models::ModelNodeInstance::Render() [virtual, inherited]

perform rendering

The Render() method is called when the ModelNodeInstance needs to render itself. There will always be a call to the Apply() method before Render() is called, however, Render() may be called several times per Apply() invocation.

Reimplemented in Characters::CharacterSkinNodeInstance, Models::ShapeNodeInstance, and Particles::ParticleSystemNodeInstance.

bool Models::ModelNodeInstance::IsVisible() const [inherited]

return true if node instance is set to visible

FIXME: The recursion in this is method makes it slow, especially in deep hierarchies. Also it can't be inlined.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Models::StreamModelLoader
Models::StreamModelLoader Class Reference

#include <streammodelloader.h>

Inheritance diagram for Models::StreamModelLoader:
Detailed Description

A ResourceLoader class for setting up Models from Streams. Supports Nebula3 binary and XML formats, and the legacy Nebula3 nvx2 format. Relies on StreamReader classes which implement the actual stream parsing.

(C) 2007 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StreamModelLoader ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~StreamModelLoader ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>virtual bool CanLoadAsync () const</code></td>
<td>return true if asynchronous loading is supported</td>
</tr>
<tr>
<td><code>virtual bool OnLoadRequested ()</code></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td><code>virtual void OnLoadCancelled ()</code></td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td><code>virtual bool OnPending ()</code></td>
<td>call frequently while after <code>OnLoadRequested()</code> to put <code>Resource</code> into loaded state</td>
</tr>
<tr>
<td><code>void SetStreamMeshLoader (const Ptr&lt; CoreGraphics::StreamMeshLoader &gt; &amp;meshLoader)</code></td>
<td>set optional stream mesh loader, which is used for <code>shapenode</code>'s mesh loading</td>
</tr>
<tr>
<td><code>virtual void OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</code></td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td><code>virtual void OnRemoveFromResource ()</code></td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td><code>bool IsAttachedToResource () const</code></td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td><code>const Ptr&lt; Resource &gt; &amp; GetResource () const</code></td>
<td>get pointer to resource</td>
</tr>
<tr>
<td><code>Resource::State GetState () const</code></td>
<td>return current state</td>
</tr>
<tr>
<td><code>virtual void Reset ()</code></td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>increment refcount by one</code></td>
<td></td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName()</code> const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC()</code> const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void DumpRefCountingLeaks()</code></td>
<td><code>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</code></td>
</tr>
</tbody>
</table>
## Protected Member Functions

```c++
void SetState (Resource::State S)
```

*set current state*
**Member Function Documentation**

```c
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```c
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```c
void Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```c
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```c
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```c
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Models::TransformNode
Models::TransformNode Class Reference

#include <transformnode.h>

Inheritance diagram for Models::TransformNode:
Detailed Description

Defines a transformation in a transform hierarchy.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TransformNode()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~TransformNode()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt; ModelNodeInstance &gt;</code></td>
<td><code>CreateNodeInstance()</code> const create a model node instance</td>
</tr>
<tr>
<td>virtual <code>bool ParseDataTag()</code></td>
<td><code>ParseDataTag</code> (const <code>Util::FourCC</code> &amp;fourCC, const <code>Ptr&lt; IO::BinaryReader &gt;</code> &amp;reader) parse data tag (called by loader code)</td>
</tr>
<tr>
<td>virtual <code>Resources::Resource::State GetResourceState()</code></td>
<td>get overall state of contained resources (Initial, Loaded, Pending, Failed, Cancelled)</td>
</tr>
<tr>
<td><code>void SetPosition(const Math::point &amp;)</code></td>
<td><code>setPosition</code></td>
</tr>
<tr>
<td><code>const Math::point &amp; GetPosition()</code></td>
<td><code>GetPosition</code></td>
</tr>
<tr>
<td><code>void SetRotation(const Math::quaternion &amp;)</code></td>
<td><code>set rotate quaternion</code></td>
</tr>
<tr>
<td><code>const Math::quaternion &amp; GetRotation()</code></td>
<td><code>get rotate quaternion</code></td>
</tr>
<tr>
<td><code>void SetScale(const Math::vector &amp;)</code></td>
<td><code>setScale</code></td>
</tr>
<tr>
<td><code>const Math::vector &amp; GetScale()</code></td>
<td><code>get scale</code></td>
</tr>
<tr>
<td><code>void SetRotatePivot(const Math::point &amp;)</code></td>
<td><code>set rotate pivot</code></td>
</tr>
<tr>
<td><code>const Math::point &amp; GetRotatePivot()</code></td>
<td><code>get rotate pivot</code></td>
</tr>
<tr>
<td><code>void SetScalePivot(const Math::point &amp;)</code></td>
<td><code>setScale pivot</code></td>
</tr>
<tr>
<td><code>const Math::point &amp;</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetScalePivot()</code> const</td>
<td>get scale pivot</td>
</tr>
<tr>
<td><code>IsInViewSpace()</code> const</td>
<td>is transformnode in viewspace</td>
</tr>
<tr>
<td><code>SetInViewSpace(bool b)</code></td>
<td>set transformnode in viewspace</td>
</tr>
<tr>
<td><code>GetMinDistance()</code> const</td>
<td>get MinDistance</td>
</tr>
<tr>
<td><code>SetMinDistance(float val)</code></td>
<td>set MinDistance</td>
</tr>
<tr>
<td><code>GetMaxDistance()</code> const</td>
<td>get MaxDistance</td>
</tr>
<tr>
<td><code>SetMaxDistance(float val)</code></td>
<td>set MaxDistance</td>
</tr>
<tr>
<td><code>LodDistancesUsed()</code> const</td>
<td>are lod distances used</td>
</tr>
<tr>
<td><code>OnAttachToModel(const Ptr&lt; I &gt;&amp; model)</code></td>
<td>called when attached to model node</td>
</tr>
<tr>
<td><code>GetLockedToViewer()</code> const</td>
<td>get LockedToViewer</td>
</tr>
<tr>
<td><code>SetLockedToViewer(bool val)</code></td>
<td>set LockedToViewer</td>
</tr>
<tr>
<td><code>CheckLodDistance(float distToViewer)</code></td>
<td>helper method to check whether the distance within lod distances</td>
</tr>
<tr>
<td><code>CreateNodeInstanceHierarchy(Ptr&lt; ModelNodeInstance &gt; &amp;modelInst)</code></td>
<td>recursively create model node instance and model node instances</td>
</tr>
<tr>
<td><code>OnRemoveFromModel()</code></td>
<td>called when removed from model node</td>
</tr>
<tr>
<td><code>LoadResources()</code></td>
<td>called when resources should be loaded</td>
</tr>
<tr>
<td><code>UnloadResources()</code></td>
<td>called when resources should be unloaded</td>
</tr>
</tbody>
</table>
virtual void OnResourcesLoaded ()
called once when all pending resource have loaded

virtual void BeginParseDataTags ()
begin parsing data tags

virtual void EndParseDataTags ()
finish parsing data tags

virtual void ApplySharedState (IndexT frameIndex)
apply state shared by all my ModelNodeInstances

bool IsAttachedToModel () const
return true if currently attached to a Model

const Ptr< Model > & GetModel () const
get model this node is attached to

void SetResourceStreamingLevelOfDetail (float factor)
sets the resourceStreamingLevelOfDetail but only if the given value is bigger than the current one (reseted on frame-start)

void ResetScreenSpaceStats ()
resets all screen space stats e.g. size

void SetBoundingBox (const Math::bbox &b)
set bounding box

const Math::bbox & GetBoundingBox () const
get bounding box of model node

void SetName (const Util::StringAtom &)
set model node name

const Util::StringAtom & GetName () const
get model node name

void SetType (ModelNodeType::Code)
set ModelNodeType

ModelNodeType::Code GetType () const
get the ModelNodeType

void SetParent (const Ptr< ModelNode >&p)
set parent node

const Ptr< ModelNode > & GetParent () const
bool HasParent () const
return true if node has a parent

void AddChild (const Ptr< ModelNode > &c)
add a child node

const Util::Array< Ptr< ModelNode > > & GetChildren () const
get child nodes

bool HasChild (const Util::StringAtom &name) const
return true if a direct child exists by name

const Ptr< ModelNode > & LookupChild (const Util::StringAtom &name) const
get pointer to direct child by name

void AddVisibleNodeInstance (IndexT frameIndex, const Ptr< ModelNodeInstance > &nodeInst)
called by model node instance on NotifyVisible()

const Util::Array< Ptr< ModelNodeInstance > > & GetVisibleModelNodeInstances (ModelNodeType::Code t) const
get visible model node instances

bool HasStringAttr (const Util::StringAtom &attrId) const
has string attr

const Util::StringAtom & GetStringAttr (const Util::StringAtom &attrId) const
get string attr

void SetStringAttr (const Util::StringAtom &attrId, const Util::StringAtom &value)
add string attribute

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is
### bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class.

### bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string.

### bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc.

### bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class.

### bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string.

### bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc.

### const Util::String & GetClassName () const
get the class name.

### Util::FourCC GetClassFourCC () const
get the class FourCC code.
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual Ptr&lt; ModelNodeInstance &gt; RecurseCreateNodeInstanceHierarchy(const Ptr&lt; ModelInstance &gt;&amp; modelInst, const Ptr&lt; ModelNodeInstance &gt;&amp; parentNodeInst)</code></td>
<td>recursively create node instance hierarchy</td>
</tr>
</tbody>
</table>
## Protected Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>resourceStreamingLevelOfDetail</code></td>
<td>Factor between 0.0 (close) and 1.0 (far away) describing the distance to camera (used for decision of max needed mipMap)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
Ptr< ModelNodeInstance > Models::ModelNode::CreateNodeInstanceHierarchy ( const Ptr< ModelInstance > & modelInst ) [inherited]
```

recursively create model node instance and child model node instances

Create the node instance hierarchy.

```cpp
void Models::ModelNode::LoadResources ( ) [virtual, inherited]
```

called when resources should be loaded

This method is called when required resources should be loaded.

Reimplemented in Characters::CharacterNode, Models::ShapeNode, Models::StateNode, and Particles::ParticleSystemNode.

```cpp
void Models::ModelNode::UnloadResources ( ) [virtual, inherited]
```

called when resources should be unloaded

This method is called when required resources should be unloaded.

Reimplemented in Characters::CharacterNode, Models::ShapeNode, Models::StateNode, and Particles::ParticleSystemNode.

```cpp
void Models::ModelNode::OnResourcesLoaded ( ) [virtual, inherited]
```

called once when all pending resource have been loaded

This method is called once by Model::OnResourcesLoaded() when all pending resources of a model have been loaded.
Reimplemented in **Characters::CharacterNode**, and **Particles::ParticleSystemNode**.

```cpp
void Models::ModelNode::BeginParseDataTags() [virtual, inherited]
```

begin parsing data tags

Begin parsing data tags. This method is called by **StreamModelLoader** before **ParseDataTag()** is called for the first time.

```cpp
void Models::ModelNode::EndParseDataTags() [virtual, inherited]
```

finish parsing data tags

End parsing data tags. This method is called by **StreamModelLoader** after the last **ParseDataTag()** is called.

```cpp
void Models::ModelNode::ApplySharedState(IndexT frameIndex) [virtual, inherited]
```

apply state shared by all my ModelNodeInstances

This method is called once before rendering the ModelNode's visible instance nodes through the **ModelNodeInstance::ApplyState()** and **ModelNodeInstance::Render()** methods. The method must apply the shader state that is shared across all instances. Since this state is constant across all instance nodes, this only happens once before rendering an instance set.

Reimplemented in **Characters::CharacterSkinNode**, **Models::ShapeNode**, and **Models::StateNode**.

```cpp
void Models::ModelNode::ResetScreenSpaceStats() [inline, inherited]
```

resets all screen space stats e.g. size

Reset resourceStreamingLevelOfDetail to -1.0 as we are able to
recognize invisible items this way. (visible items will overwrite this value with a value >= 0.0)

```cpp
Ptr< ModelNodeInstance >
Models::ModelNode::RecurseCreateNodeInstanceHierarchy ( const Ptr< ModellInstance > & modelInst,
const Ptr< ModelNodeInstance > & parentNodeInst )
```

recursively create node instance hierarchy

Recursively create node instances and attach them to the provided model instance. Returns a pointer to the root node instance.

```cpp
int
Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void
Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.
get the class FourCC code

Get the class FourCC of the object.

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Models::TransformNodeInstance
#include <transformnodeinstance.h>

Inheritance diagram for Models::TransformNodeInstance:
Detailed Description

Holds and applies per-node-instance transformation.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TransformNodeInstance()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~TransformNodeInstance()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>OnRenderBefore(IndexT frameIndex, Timing::Time time)</code></td>
<td>called from <code>ModelEntity::OnRenderBefore</code></td>
</tr>
<tr>
<td><code>ApplyState()</code></td>
<td>apply per-instance state prior to rendering</td>
</tr>
<tr>
<td><code>setPosition(const Math::point &amp;p)</code></td>
<td>set position</td>
</tr>
<tr>
<td><code>GetPosition()</code></td>
<td>get position</td>
</tr>
<tr>
<td><code>rotate(const Math::quaternion &amp;r)</code></td>
<td>set rotate quaternion</td>
</tr>
<tr>
<td><code>GetRotate()</code></td>
<td>get rotate quaternion</td>
</tr>
<tr>
<td><code>setScale(const Math::vector &amp;s)</code></td>
<td>set scale</td>
</tr>
<tr>
<td><code>getScale()</code></td>
<td>get scale</td>
</tr>
<tr>
<td><code>setRotatePivot(const Math::point &amp;p)</code></td>
<td>set rotate pivot</td>
</tr>
<tr>
<td><code>getRotatePivot()</code></td>
<td>get rotate pivot</td>
</tr>
<tr>
<td><code>setScalePivot(const Math::point &amp;p)</code></td>
<td></td>
</tr>
</tbody>
</table>
const Math::point & GetScalePivot() const
get scale pivot

void SetOffsetMatrix(const Math::matrix44 &m)
set optional offset matrix

const Math::matrix44 & GetOffsetMatrix() const
get optional offset matrix

bool IsInViewSpace() const
is transformnode in viewspace

void SetInViewSpace(bool b)
set transformnode in viewspace

bool GetLockedToViewer() const
get LockedToViewer

void SetLockedToViewer(bool val)
set LockedToViewer

const Math::matrix44 & GetLocalTransform()
get resulting local transform matrix in local parent space

const Math::matrix44 & GetModelTransform() const
get model space transform (valid after Update())

virtual void DiscardHierarchy()
discard the model node instance and all of its children

bool IsValid() const
return true if the model node instance is valid

virtual void OnNotifyCullingVisible(IndexT frameIndex, Timing::Time time)
called from ModelEntity::OnNotifyCullingVisible

virtual void OnVisibilityResolve(IndexT resolveIndex, float distanceToViewer)
called during visibility resolve
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Render ()</td>
<td>perform rendering</td>
</tr>
<tr>
<td>GetName () const</td>
<td>get model node name</td>
</tr>
<tr>
<td>HasParent () const</td>
<td>return true if node has a parent</td>
</tr>
<tr>
<td>GetParent () const</td>
<td>get parent node</td>
</tr>
<tr>
<td>GetChildren () const</td>
<td>get child nodes</td>
</tr>
<tr>
<td>HasChild (const Util::StringAtom &amp;name) const</td>
<td>return true if a direct child exists by name</td>
</tr>
<tr>
<td>LookupChild (const Util::StringAtom &amp;name) const</td>
<td>get pointer to direct child by name</td>
</tr>
<tr>
<td>LookupPath (const Util::String &amp;path)</td>
<td>get modelnodeinstance by hierarchy path</td>
</tr>
<tr>
<td>GetModellInstance () const</td>
<td>get the ModellInstance we are attached to</td>
</tr>
<tr>
<td>GetModelNode () const</td>
<td>get the ModelNode we're associated with</td>
</tr>
<tr>
<td>SetVisible (bool b, Timing::Time time, bool recursive=true)</td>
<td>set the node instance's visibility</td>
</tr>
<tr>
<td>IsVisible () const</td>
<td>return true if node instance is set to visible</td>
</tr>
<tr>
<td>GetModelNodeInstanceIndex () const</td>
<td>get model node instance index for</td>
</tr>
<tr>
<td>GetModelNodeIndex () const</td>
<td>get model node instance index for</td>
</tr>
<tr>
<td>GetModelNode () const</td>
<td>get the ModelNode we're associated with</td>
</tr>
</tbody>
</table>
int **GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & Get ClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>Setup</strong> (const <strong>Ptr</strong>&lt; <strong>ModelInstance</strong> &gt; &amp;inst, const <strong>Ptr</strong>&lt; <strong>ModelNode</strong> &gt; &amp;node, const <strong>Ptr</strong>&lt; <strong>ModelNodeInstance</strong> &gt; &amp;parentNodeInst)</td>
<td>called when attached to <strong>ModelInstance</strong></td>
</tr>
<tr>
<td>virtual void <strong>Discard</strong> ()</td>
<td>called when removed from <strong>ModelInstance</strong></td>
</tr>
<tr>
<td>virtual void <strong>RenderDebug</strong> ()</td>
<td>render node specific debug shape</td>
</tr>
<tr>
<td>void <strong>SetParent</strong> (const <strong>Ptr</strong>&lt; <strong>ModelNodeInstance</strong> &gt; &amp;p)</td>
<td>set parent node</td>
</tr>
<tr>
<td>void <strong>AddChild</strong> (const <strong>Ptr</strong>&lt; <strong>ModelNodeInstance</strong> &gt; &amp;c)</td>
<td>add a child node</td>
</tr>
<tr>
<td>virtual void <strong>OnShow</strong> (<strong>Timing::Time</strong> time)</td>
<td>called when the node becomes visible with current time</td>
</tr>
<tr>
<td>virtual void <strong>OnHide</strong> (<strong>Timing::Time</strong> time)</td>
<td>called when the node becomes invisible</td>
</tr>
</tbody>
</table>
Member Function Documentation

```c++
void Models::TransformNodeInstance::OnRenderBefore(IndexT frameIndex, Timing::Time time) [virtual]
called from ModelEntity::OnRenderBefore

The update method should first invoke any animators which change per-instance attributes (this is done in the parent class). Then the local space transforms must be flattened into model space.

NOTE: this method must be called late in the frame to give other systems a chance to modify the transform matrix (for instance the character attachment system).

Reimplemented from Models::ModelNodeInstance.
Reimplemented in Particles::ParticleSystemNodeInstance.
```

```c++
void Models::TransformNodeInstance::ApplyState() [virtual]
apply per-instance state prior to rendering

Set our model matrix (computed in the Update() method) as current model matrix in the TransformDevice.

Reimplemented from Models::ModelNodeInstance.
Reimplemented in Models::StateNodeInstance, and Particles::ParticleSystemNodeInstance.
```

```c++
void Models::TransformNodeInstance::RenderDebug() [protected, virtual]
render node specific debug shape

Render a debug visualization of the node.
```
Reimplemented from **Models::ModelNodeInstance**.

Reimplemented in **Characters::CharacterNodeInstance**, and **Particles::ParticleSystemNodeInstance**.

```cpp
void Models::ModelNodeInstance::DiscardHierarchy() [virtual, inherited]
```

discard the model node instance and all of its children

Discards this model node instance and all of its children recursively.

```cpp
void Models::ModelNodeInstance::Render() [virtual, inherited]
```

perform rendering

The **Render()** method is called when the **ModelNodeInstance** needs to render itself. There will always be a call to the Apply() method before **Render()** is called, however, **Render()** may be called several times per Apply() invocation.

Reimplemented in **Characters::CharacterSkinNodeInstance**, **Models::ShapeNodeInstance**, and **Particles::ParticleSystemNodeInstance**.

```cpp
bool Models::ModelNodeInstance::IsVisible() const [inherited]
```

return true if node instance is set to visible

FIXME: The recursion in this is method makes it slow, especially in deep hierarchies. Also it can't be inlined.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
( ) [inline, inherited]
```
Core::RefCounted::AddRef

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Models::VisResolveContainer
Models::VisResolveContainer< TYPE >
Class Template Reference

#include <visresolvecontainer.h>
Detailed Description

template<class TYPE>
class Models::VisResolveContainer< TYPE >

Helper class which keeps an array of visible nodes by type.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VisResolveContainer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>Reset ()</strong></td>
<td>Reset content</td>
</tr>
<tr>
<td>void <strong>SetResolved (ModelNodeType::Code t, bool b)</strong></td>
<td>Set the resolved flag for a given ModelNodeType</td>
</tr>
<tr>
<td>bool <strong>IsResolved (ModelNodeType::Code t)</strong> const</td>
<td>Return true if the resolved flag has been set</td>
</tr>
<tr>
<td>void <strong>Add (IndexT frameIndex, ModelNodeType::Code t, const Ptr&lt;TYPE&gt; &amp;e)</strong></td>
<td>Add a visible element by ModelNodeType</td>
</tr>
<tr>
<td>const <strong>Util::Array&lt; Ptr&lt;TYPE&gt; &gt; &amp; Get (ModelNodeType::Code t)</strong> const</td>
<td>Get all visible elements of given ModelNodeType</td>
</tr>
</tbody>
</table>
Models::VisResolver
Models::VisResolver Class Reference

#include <visresolver.h>

Inheritance diagram for Models::VisResolver:
Detailed Description

The VisResolver accepts visible ModelInstances and resolves them into their ModelNodeInstances, organized into node type and sorted for optimal rendering (instances of the same ModelNode should be rendered together to reduce state switch overhead).

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Public</strong></td>
<td></td>
</tr>
<tr>
<td><code>VisResolver ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~VisResolver ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>Open ()</code></td>
<td>open the visibility resolver</td>
</tr>
<tr>
<td><code>Close ()</code></td>
<td>close the visibility resolver</td>
</tr>
<tr>
<td><code>IsOpen () const</code></td>
<td>return true if currently open</td>
</tr>
<tr>
<td><code>BeginResolve (const Math::math &amp;cameraTransform)</code></td>
<td>begin resolving</td>
</tr>
<tr>
<td><code>AttachVisibleModelInstance (const ModelInstance &amp;inst)</code></td>
<td>attach a visible ModelInstance</td>
</tr>
<tr>
<td><code>AttachVisibleModelInstancePlayerCamera (const Ptr&lt;ModelInstance&gt; &amp;inst)</code></td>
<td>attach a visible ModelInstance</td>
</tr>
<tr>
<td><code>EndResolve ()</code></td>
<td>end resolve</td>
</tr>
<tr>
<td><code>GetVisibleModels (ModelNodeType::nodeType) const</code></td>
<td>post-resolve: get Models with visible ModelNode instances by node type</td>
</tr>
<tr>
<td><code>GetVisibleModelNodes (ModelNodeType::Code nodeType, const Ptr&lt;Model&gt; &amp;model) const</code></td>
<td>post-resolve: get visible ModelNodes of a Model type</td>
</tr>
<tr>
<td><code>GetVisibleModelNodeInstance (ModelNodeType::Code nodeType, const Ptr&lt;ModelNode&gt; &amp;modelNode)</code></td>
<td>post-resolve: get visible ModelNodeInstance of a ModelNode by node type</td>
</tr>
</tbody>
</table>
int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class or derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class or derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class or derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

int
Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Net::DebugMessage
Net::DebugMessage Class Reference

#include <debugmessage.h>
Detailed Description

Encapsulates a stream and a port number for debug communication. Currently this class is only used on the Wii for communication over the HIO2 channel, but is public for the PC proxy tools communicating with the Wii devkit.

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DebugMessage ()</strong></td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><strong>DebugMessage (ushort port, const Ptr&lt; IO::Stream &gt; &amp;data)</strong></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td><code>ushort GetPort () const</code></td>
<td><em>get the port number</em></td>
</tr>
<tr>
<td><code>const Ptr&lt; IO::Stream &gt; &amp; GetStream () const</code></td>
<td><em>get the data stream</em></td>
</tr>
<tr>
<td><code>bool IsValid () const</code></td>
<td><em>return true if the message is valid</em></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:48 2010
Net::DebugPacket
Net::DebugPacket Class Reference

#include <debugpacket.h>
Detailed Description

Encapsulates a data packet for debug communication. Currently this class is only used on the Wii to communicate on the host PC, with the Wii specific classes hidden away in the Wii port of N3. DebugPacket is "public" because it's also used in Win32 proxy tools.

Every packet consists of a 4-byte header, and 1020 bytes of payload data, so that each packet is exactly 1 KByte long.

A packet looks like this:

uint magic; // FourCC('DPKT') uint count; // message counter ushort port; // port number ushort payloadSize; // 0xFFFF if packet is full and more data follows, else size of data in packet ubyte payload[PacketSize - HeaderSize]

The payloadSize must be interpreted like this:

0xFFFF: full payload of data, current message is continued in next package else: payloadSize is number of bytes in packet, current message is complete

IMPORTANT NOTE:

The DebugPacket does not allocate any external memory, and thus can be safely used from Wii interrupt handlers.

(C) 2009 Radon Labs GmbH
Public Types

enum Command

packet commands
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DebugPacket ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~DebugPacket ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>*<em>SizeT Write (ushort portNum, uint packetCounter, constubyte <em>buf, SizeT numBytes)</em></em></td>
<td>write data to packet, returns number of bytes written, all data must be written in a single call!</td>
</tr>
<tr>
<td>*<em>void WriteRaw (constvoid <em>buf, SizeT bufSize)</em></em></td>
<td>write raw data to packet (must have valid packet header, size of src buffer must be PacketSize!)</td>
</tr>
<tr>
<td><strong>void SetDataValid (bool b)</strong></td>
<td>set the &quot;has data&quot; flag on the packet</td>
</tr>
<tr>
<td><strong>bool HasData () const</strong></td>
<td>return true if the packet contains data</td>
</tr>
<tr>
<td><strong>Util::FourCC GetMagic () const</strong></td>
<td>get magic code at start of header</td>
</tr>
<tr>
<td><strong>uint GetCount () const</strong></td>
<td>get message counter</td>
</tr>
<tr>
<td><strong>ushort GetPort () const</strong></td>
<td>get the port number of the packet</td>
</tr>
<tr>
<td><strong>bool IsFinalPacket () const</strong></td>
<td>return true if this is the last packet of a message</td>
</tr>
<tr>
<td><strong>SizeT GetPayloadSize () const</strong></td>
<td>get actual payload size</td>
</tr>
<tr>
<td><strong>constubyte * GetPayload () const</strong></td>
<td>get pointer to payload</td>
</tr>
<tr>
<td><strong>constubyte * GetRawBuffer () const</strong></td>
<td>get pointer to raw packet buffer (contains header + payload)</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EncodeStream</strong> (ushort portNum, uint firstPacketCounter, const Ptr<a href="">IO::Stream</a> &amp;stream)</td>
<td>encode stream into a series of packets</td>
</tr>
<tr>
<td><strong>DecodePackets</strong> (const Util::Array&lt;DebugPacket&gt; &amp;packets, const Ptr<a href="">IO::Stream</a> &amp;stream, ushort &amp;outPortNum)</td>
<td>decode a series of packets into a stream</td>
</tr>
</tbody>
</table>
### Static Public Attributes

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>static const</td>
<td>PacketSize</td>
<td><code>512</code></td>
</tr>
<tr>
<td></td>
<td><code>packet size</code></td>
<td></td>
</tr>
<tr>
<td>static const</td>
<td>HeaderSize</td>
<td><code>12</code></td>
</tr>
<tr>
<td></td>
<td><code>header size</code></td>
<td></td>
</tr>
<tr>
<td>static const</td>
<td>MaxPayloadSize</td>
<td><code>PacketSize - HeaderSize</code></td>
</tr>
<tr>
<td></td>
<td><code>max payload size</code></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

SizeT
Net::DebugPacket::Write(ushort portNum,
uint packetCount,
const ubyte *buf,
SizeT numBytes)

Write data to packet, returns number of bytes written, all data must be written in a single call!

Write data to the packet. Returns the number of data written. If all data could be written, the returned number is identical with the requested size, and the packet's payload size will be set to that size. This means the packet completes a message. If not all data could be written, the actually written size (MaxPayloadSize) will be returned, and the internal payloadSize will be set to 0xFFFF, meaning that the packet contains incomplete data which will be continued with the next packet.

void
Net::DebugPacket::WriteRaw( const void *buf,
SizeT bufSize)

Write raw data to packet (must have valid packet header, size of src buffer must be PacketSize!)

Write raw data to the packet. The source buffer must contain a valid packet header and the size of the source data must equal DebugPacket::PacketSize!

FourCC
Net::DebugPacket::GetMagic()

get magic code at start of header

Get the magic code at the beginning of the message. Must be
FourCC('DPKT').

uint
Net::DebugPacket::GetCount() const

get message counter

Get the packet counter of this packet.

ushort
Net::DebugPacket::GetPort() const

get the port number of the packet

Get the port number from the packet.

bool
Net::DebugPacket::IsFinalPacket() const

return true if this is the last packet of a message

Return true if this is the final packet in a multi-packet message.

SizeT
Net::DebugPacket::GetPayloadSize() const

get actual payload size

Returns the number of payload data bytes in the packet. This will always return a valid number, even if the payloadSize header member is set to the special 0xFFFF code.

const ubyte *
Net::DebugPacket::GetPayload() const

get pointer to payload

Get a pointer to the actual payload data.

const ubyte *
Net::DebugPacket::GetRawBuffer() const
get pointer to raw packet buffer (contains header + payload)

Get a pointer to the raw packet data.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**Net::IpAddress**
Net::IpAddress Class Reference

#include <ipaddress.h>
Detailed Description

Represents an IP address, consisting of a IPv4 host address and a port number. Can extract address information from an URI and automatically converts host names to addresses, and offers the special hostnames "localhost", "any", "broadcast", "self" and "inetself" where:

- "localhost" will translate to 127.0.0.1
- "any" will translate to INADDR_ANY, which is 0.0.0.0
- "broadcast" will translate to INADDR_BROADCAST, which is 255.255.255.255
- "self" will translate to the first valid tcp/ip address for this host (there may be more then one address bound to the host)
- "inetself" will translate to the first host address which is not a LAN address (which is not a class A, B, or C network) if none such exists the address will fall back to "self"

(C) 2006 Radon Labs GmbH
Net::MessageClient
Net::MessageClient Class Reference

#include <messageclient.h>

Inheritance diagram for Net::MessageClient:
Detailed Description

Wrapper class for the **Net::TcpClient** that sends data in special message container.

(C) 2009 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MessageClient ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~MessageClient ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual Result <strong>Connect ()</strong></td>
<td>establish a connection with the server</td>
</tr>
<tr>
<td>virtual void <strong>Disconnect ()</strong></td>
<td>disconnect from the server</td>
</tr>
<tr>
<td>virtual bool <strong>Send ()</strong></td>
<td>send accumulated content of send stream to server</td>
</tr>
<tr>
<td>virtual const <strong>Ptr&lt; IO::Stream &gt;</strong> &amp; <strong>GetSendStream ()</strong></td>
<td>access to send stream</td>
</tr>
<tr>
<td>virtual bool <strong>Recv ()</strong></td>
<td>receive data from server into recv stream</td>
</tr>
<tr>
<td>virtual const <strong>Ptr&lt; IO::Stream &gt;</strong> &amp; <strong>GetRecvStream ()</strong></td>
<td>access to recv stream</td>
</tr>
<tr>
<td>void <strong>SetBlocking (bool b)</strong></td>
<td>enable/disable blocking behaviour</td>
</tr>
<tr>
<td>bool <strong>IsBlocking ()</strong> const</td>
<td>get blocking behaviour</td>
</tr>
<tr>
<td>void <strong>SetServerAddress (const IpAddress &amp;addr)</strong></td>
<td>set the server address to connect to</td>
</tr>
<tr>
<td>const IpAddress &amp; <strong>GetServerAddress ()</strong> const</td>
<td>get the server address</td>
</tr>
<tr>
<td>bool <strong>IsConnected ()</strong></td>
<td>return true if currently connected</td>
</tr>
<tr>
<td>int <strong>GetRefCount ()</strong> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

TcpClient::Result
Net::MessageClient::Connect() [virtual]

establish a connection with the server

Establish a connection with the server. See TcpClient for details.

Reimplemented from Net::StdTcpClient.

bool
Net::StdTcpClient::IsConnected() [inherited]

return true if currently connected

Return true if the socket is currently connected. This will actually probe the connection using a select().

int
Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Net::MessageClientConnection
Net::MessageClientConnection Class Reference

#include <messageclientconnection.h>

Inheritance diagram for Net::MessageClientConnection:
Detailed Description

A wrapper class for the `Net::TcpClientConnection`.

All data that is send by using `Send()` will streamed into a container, before it will send to the client. The message contains information about the full length of the data. This way the receiving site can handle messages that are splittet into chunks and may wait until a full message was received.

The `MessageClientConnection` will concatenate incoming data chunks to full messages and `Recv()` will only return finished messages.

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MessageClientConnection</strong> ()</td>
<td></td>
</tr>
<tr>
<td><strong>~MessageClientConnection</strong> ()</td>
<td></td>
</tr>
<tr>
<td><strong>virtual bool Connect (const Ptr&lt; Socket &gt;&amp; s)</strong></td>
<td>connect using provided socket</td>
</tr>
<tr>
<td><strong>virtual void Shutdown ()</strong></td>
<td>shutdown the connection</td>
</tr>
<tr>
<td><strong>virtual Socket::Result Send ()</strong></td>
<td>send accumulated content of send stream to server</td>
</tr>
<tr>
<td><strong>virtual Socket::Result Send (const Ptr&lt; IO::Stream &gt;&amp; &amp;stream)</strong></td>
<td>directly send a stream to the server, often prevents a memory copy</td>
</tr>
<tr>
<td><strong>virtual const Ptr&lt; IO::Stream &gt;&amp; GetSendStream ()</strong></td>
<td>access to send stream</td>
</tr>
<tr>
<td><strong>virtual Socket::Result Recv ()</strong></td>
<td>receive data from server into recv stream</td>
</tr>
<tr>
<td><strong>virtual const Ptr&lt; IO::Stream &gt;&amp; GetRecvStream ()</strong></td>
<td>access to recv stream</td>
</tr>
<tr>
<td><strong>bool IsConnected () const</strong></td>
<td>get the connection status</td>
</tr>
<tr>
<td><strong>const IpAddress &amp; GetClientAddress () const</strong></td>
<td>get the client's ip address</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsInstanceOf</code> (const <code>Util::FourCC</code> &amp;classFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><code>return true if this object is instance of given class by fourcc</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><code>IsA</code> (const <code>Rtti</code> &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>return true if this object is instance of given class, or a derived class</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><code>IsA</code> (const <code>Util::String</code> &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>return true if this object is instance of given class, or a derived class, by string</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>return true if this object is instance of given class, or a derived class, by fourcc</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <code>Util::String</code> &amp;</th>
<th><code>GetClassName</code> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>get the class name</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>Util::FourCC</code></th>
<th><code>GetClassFourCC</code> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>get the class FourCC code</code></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Constructor & Destructor Documentation

Net::MessageClientConnection::MessageClientConnection ( )

Constructor

Net::MessageClientConnection::~MessageClientConnection ( ) [virtual]

Destructor
Member Function Documentation

bool Net::MessageClientConnection::Connect(const Ptr<Socket>& s) [virtual]

connect using provided socket

Connect using provided socket. See `TcpClientConnection` for details.

Reimplemented from `Net::StdTcpClientConnection`.

void Net::MessageClientConnection::Shutdown() [virtual]

shutdown the connection

This disconnects the current connection.

Reimplemented from `Net::StdTcpClientConnection`.

Socket::Result Net::MessageClientConnection::Send() [virtual]

send accumulated content of send stream to server

Writes accumulated send stream into a message container and sends
this container.

Reimplemented from `Net::StdTcpClientConnection`.

Socket::Result Net::MessageClientConnection::Send(const Ptr<IO::Stream>& stream) [virtual]

directly send a stream to the server, often prevents a memory copy

Writes given stream into a message container and sends this container.
Reimplemented from **Net::StdTcpClientConnection**.

```cpp
const Ptr<IO::Stream> & Net::MessageClientConnection::GetSendStream() [virtual]
```

access to send stream

Returns the stream to which data for sending is written.

Reimplemented from **Net::StdTcpClientConnection**.

```cpp
Socket::Result Net::MessageClientConnection::Recv() [virtual]
```

receive data from server into recv stream

Receive data from the clients, but returns only Success, if a complete message was available. This message will either return one complete message, or none at all.

Reimplemented from **Net::Std TcpClientConnection**.

```cpp
const Ptr<IO::Stream> & Net::MessageClientConnection::GetRecvStream() [virtual]
```

access to recv stream

Returns the stream with the received data

Reimplemented from **Net::StdTcpClientConnection**.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one
Increment the refcount of the object.

```cpp
void Core::RefCounted::Release( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Net::Socket
#include <socket.h>

Inheritance diagram for Net::Socket:
Detailed Description

Platform independent wrapper class for the Sockets API.

(C) 2006 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Protocol</th>
<th>protocol types</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>ErrorCode</td>
<td>error codes</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool Open (Protocol p)</code></td>
<td>open the socket</td>
</tr>
<tr>
<td><code>void Close ()</code></td>
<td>close the socket</td>
</tr>
<tr>
<td><code>bool IsOpen () const</code></td>
<td>return true if the socket is open</td>
</tr>
<tr>
<td><code>ErrorCode GetErrorCode () const</code></td>
<td>get the last error code</td>
</tr>
<tr>
<td><code>Util::String GetErrorString () const</code></td>
<td>get the last error string</td>
</tr>
<tr>
<td><code>void SetAddress (const Net::IpAddress &amp;a)</code></td>
<td>set internet address of socket</td>
</tr>
<tr>
<td><code>const Net::IpAddress &amp; GetAddress () const</code></td>
<td>get internet address of socket</td>
</tr>
<tr>
<td><code>void SetBroadcast (bool b)</code></td>
<td>set the broadcast flag (SO_BROADCAST)</td>
</tr>
<tr>
<td><code>bool GetBroadcast ()</code></td>
<td>get the broadcast flag</td>
</tr>
<tr>
<td><code>void SetKeepAlive (bool b)</code></td>
<td>set the keepalive flag (SO_KEEPALIVE)</td>
</tr>
<tr>
<td><code>bool GetKeepAlive ()</code></td>
<td>get the keepalive flag</td>
</tr>
<tr>
<td><code>void SetReUseAddr (bool b)</code></td>
<td>set reuseaddr flag (SO_REUSEADDR)</td>
</tr>
<tr>
<td><code>bool GetReUseAddr ()</code></td>
<td>get reuseaddr flag</td>
</tr>
<tr>
<td><code>void SetNoDelay (bool b)</code></td>
<td>set nodelay flag (TCP_NODELAY)</td>
</tr>
<tr>
<td><code>bool GetNoDelay ()</code></td>
<td>get nodelay flag</td>
</tr>
<tr>
<td><code>void SetRecvBufSize (SizeT s)</code></td>
<td>set receive buffer size</td>
</tr>
<tr>
<td><code>SizeT GetRecvBufSize ()</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>SetSendBufSize(SizeT s)</code></td>
<td>set send buffer size</td>
</tr>
<tr>
<td><code>GetSizeSendBufSize()</code></td>
<td>get send buffer size</td>
</tr>
<tr>
<td><code>SetBlocking(bool b)</code></td>
<td>set blocking mode (FIONBIO)</td>
</tr>
<tr>
<td><code>GetBlocking()</code></td>
<td>get blocking mode</td>
</tr>
<tr>
<td><code>GetMaxMsgSize()</code></td>
<td>get the maximum message size that can be sent atomically</td>
</tr>
<tr>
<td><code>Bind()</code></td>
<td>bind socket to ip address</td>
</tr>
<tr>
<td><code>IsBound()</code></td>
<td>return true if the socket is bound to an address</td>
</tr>
<tr>
<td><code>Listen()</code></td>
<td>listen for incoming connections (for server sockets)</td>
</tr>
<tr>
<td><code>Accept(Ptr&lt;Net::Socket&gt; &amp;outSocket)</code></td>
<td>accept incoming connection, return a new socket (for server sockets)</td>
</tr>
<tr>
<td><code>Connect()</code></td>
<td>connect to the sockets address (for client sockets)</td>
</tr>
<tr>
<td><code>IsConnected()</code></td>
<td>test if the socket is currently connected</td>
</tr>
<tr>
<td><code>Send(const void* buf, SizeT numBytes, SizeT &amp;bytesSent)</code></td>
<td>send raw data into the socket</td>
</tr>
<tr>
<td><code>HasRecvData()</code></td>
<td>return true if recv data is available at the socket</td>
</tr>
<tr>
<td><code>Recv(void* buf, SizeT bufSize, SizeT &amp;bytesReceived)</code></td>
<td>receive raw data from the socket</td>
</tr>
<tr>
<td><code>SendTo(const void* buf, SizeT numBytes, uint addr, ushort port, SizeT &amp;bytesSent)</code></td>
<td>send raw data to address for connectionless sockets</td>
</tr>
<tr>
<td><code>RecvFrom(void* buf, SizeT bufSize, uint addr, ushort port, SizeT &amp;bytesReceived)</code></td>
<td>receive raw data from the socket for connectionless sockets</td>
</tr>
</tbody>
</table>
receive raw data from address for connectionless sockets

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><strong>InitNetwork</strong> ()</td>
<td><em>static initializer method (called by SysFunc::Setup())</em></td>
</tr>
<tr>
<td>static bool</td>
<td><strong>IsNetworkInitialized</strong> ()</td>
<td><em>is network initialized</em></td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Win360::Win360Socket::InitNetwork() [static, inherited]

static initializer method (called by SysFunc::Setup())

This is a one-time init for the Windows Sockets system. The method is called from SysFunc::Setup() once at startup before any threads are launched.

void
Win360::Win360Socket::SetAddress(const Net::IpAddress & a) [inline, inherited]

set internet address of socket

Set internet address of socket.

const Net::IpAddress &
Win360::Win360Socket::GetAddress() const [inline, inherited]

get internet address of socket

Get internet address of socket.

void
Win360::Win360Socket::SetBlocking(bool b) [inherited]

set blocking mode (FIONBIO)

Set the socket to blocking mode.

bool
Win360::Win360Socket::Bind() [inherited]

bind socket to ip address

Bind the socket to its ip address set with SetAddress() and SetPort(). After binding the socket to an address, call the Listen() method to wait for incoming connections. This method only makes sense for server
sockets.

```cpp
bool Win360::Win360Socket::Listen() [inherited]
```

listen for incoming connections (for server sockets)

Wait for incoming connections to a server socket. Call this method on server side after binding the socket to its address.

```cpp
bool Win360::Win360Socket::Accept(Net::Socket outSocket) [inherited]
```

accept incoming connection, return a new socket (for server sockets)

Accept an incoming connection to a server socket. This will spawn a new socket for the connection which will be returned in the provided pointer reference. The address of the returned socket will be set to the address of the "connecting entity".

```cpp
Win360Socket::Result Win360::Win360Socket::Connect() [inherited]
```

connect to the sockets address (for client sockets)

Connect to a server socket. This method is called by a client socket to connect to a server socket identified by the socket object's address. A non-blocking socket will return immediately with WouldBlock, since the connection cannot be established immediately. In this case, just continue to call `Connect()` until the method returns Success, or alternative, check the `IsConnected()` method, which will also return true once the connection has been establish.

```cpp
bool Win360::Win360Socket::IsConnected() [inherited]
```

test if the socket is currently connected

This tests if the socket is actually connected by doing a select() on the socket to probe for writability. So the `IsConnected()` method basically checks whether data can be sent through the socket.
Win360Socket::Result Win360::Win360Socket::Send ( const void* buf, 
    SizeT numBytes, 
    SizeT& bytesSent ) [inherited]

send raw data into the socket

Send raw data into the socket. Note that depending on the buffer size of the underlying socket implementation and other sockets, the method may not be able to send all provided data. In this case, the returned content of bytesSent will be less then numBytes, even though the return value will be Success. It is up to the caller to handle the extra data which hasn't been sent with the current call.

bool Win360::Win360Socket::HasRecvData ( ) [inherited]

return true if recv data is available at the socket

This method checks if the socket has received data available. Use this method in a loop with Recv() to get all data waiting at the socket. This method will never block.

Win360Socket::Result Win360::Win360Socket::Recv ( void* buf, 
    SizeT bufSize, 
    SizeT& bytesReceived ) [inherited]

receive raw data from the socket

Receive raw data from a socket and write the received data into the provided buffer. On a blocking socket this method will block until data arrives at the socket. A non-blocking socket would immediately return in this case with a WouldBlock result. When valid data has been received the method will return with a Success result and the bytesReceived argument will contain the number of received bytes. It is not guaranteed that a single receive will return all data waiting on
the socket. To make sure that the socket is really empty, call **Recv()** in a loop until **HasRecvData()** returns false. When the socket has been gracefully closed by the other side, the method will return with a Closed return value. Everything else will return with an Error return code. Call **GetErrorCode()** or **GetErrorString()** to find out more in this case.

```c++
Win360Socket::Result
Win360::Win360Socket::SendTo (const void* buf,
    SizeT numBytes,
    uint addr,
    ushort port,
    SizeT& bytesSent
) [inherited]
```

send raw data to address for connectionless sockets

**FIXME:** this is the send method for connectionless sockets using the UDP protocol.

```c++
Win360Socket::Result
Win360::Win360Socket::RecvFrom (void* buf,
    SizeT bufSize,
    uint addr,
    ushort port,
    SizeT& bytesReceived
) [inherited]
```

receive raw data from address for connectionless sockets

**FIXME:** this is the recv method for connectionless socket using the UDP protocol.

```c++
int
Core::RefCounted::GetRefCount () const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.
increment refcount by one

Increment the refcount of the object.

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

get the class name

Get the class name of the object.

get the class FourCC code

Get the class FourCC of the object.

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Net::StdTcpClient
Net::StdTcpClient Class Reference

#include <stdtcpclient.h>

Inheritance diagram for Net::StdTcpClient:
A **TcpClient** object is used to communicate with a **TcpServer**. Any number of clients can connect to a **TcpServer**, each connected client spawns a **TcpClientConnection** object on the server side which represents this client on the server. Sending and receiving data is handled through streams, streams offer the most flexible model to read and write data in different formats by connecting different stream readers and stream writers. The idea is to write data to the send stream, and to send of the accumulated data in the send stream once by calling the **Send()** method. To receive data from the server, call the **Recv()** method which will either block until, or return true in non-blocking mode as soon as data is available. The received data will be written into the receive stream, where the application can read it in any way it desires.
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StdTcpClient</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~StdTcpClient</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void SetBlocking (bool b)</strong></td>
<td>enable/disable blocking behaviour</td>
</tr>
<tr>
<td><strong>bool IsBlocking () const</strong></td>
<td>get blocking behaviour</td>
</tr>
<tr>
<td><strong>void SetServerAddress (const IpAddress &amp;addr)</strong></td>
<td>set the server address to connect to</td>
</tr>
<tr>
<td><strong>const IpAddress &amp; GetServerAddress () const</strong></td>
<td>get the server address</td>
</tr>
<tr>
<td><strong>Result Connect ()</strong></td>
<td>establish a connection with the server</td>
</tr>
<tr>
<td><strong>void Disconnect ()</strong></td>
<td>disconnect from the server</td>
</tr>
<tr>
<td><strong>bool IsConnected ()</strong></td>
<td>return true if currently connected</td>
</tr>
<tr>
<td><strong>bool Send ()</strong></td>
<td>send accumulated content of send stream to server</td>
</tr>
<tr>
<td><strong>const Ptr&lt; IO::Stream &gt; &amp; GetSendStream ()</strong></td>
<td>access to send stream</td>
</tr>
<tr>
<td><strong>bool Recv ()</strong></td>
<td>receive data from server into recv stream</td>
</tr>
<tr>
<td><strong>const Ptr&lt; IO::Stream &gt; &amp; GetRecvStream ()</strong></td>
<td>access to recv stream</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td></td>
</tr>
</tbody>
</table>
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const

return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const

return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const

return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const

return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const

return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const

get the class name

Util::FourCC GetClassFourCC () const

get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBUA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
StdTcpClient::Result
Net::StdTcpClient::Connect()
```

establish a connection with the server

Establish a connection with the server. If the client is set to non-blocking at the time this method is called, it will return immediately with the result Connecting. To check if the connection is standing, just call `Connect()` again in intervals which will eventually return Success. On a blocking client, connect returns after a connection has been established, or with a time out when no connection could be established.

Reimplemented in `Net::MessageClient`.

```cpp
void
Net::StdTcpClient::Disconnect()
```

disconnect from the server

This disconnects the current connection.

Reimplemented in `Net::MessageClient`.

```cpp
bool
Net::StdTcpClient::IsConnected()
```

return true if currently connected

Return true if the socket is currently connected. This will actually probe the connection using a select().

```cpp
int
Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Net::StdTcpClientConnection
Net::StdTcpClientConnection Class Reference

#include <stdtcpclientconnection.h>

Inheritance diagram for Net::StdTcpClientConnection:

```
Core::RefCounted

Net::StdTcpClientConnection

Net::TcpClientConnection

Net::MessageClientConnection
```
Detailed Description

A **TcpClientConnection** represents a connected **TcpClient** on the server side. **TcpClientConnection** objects are created and maintained by a **TcpServer** object over the lifetime of a client connection. **TcpClientConnection** objects are used to communicate directly with the specific client represented by the connection object.

**TcpClientConnection** objects are generally non-blocking. To receive data from the client, call the **Recv()** method until it returns true, this indicates that received data is available in the RecvStream. To read data from the RecvStream attach a StreamReader which matches the data format your expecting from the client (e.g. BinaryReader, TextReader, XmlReader, etc...). To send data back to the client just do the reverse: write data to the SendStream, and at any time call the **Send()** method which will send all data accumulated in the SendStream to the client.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th></th>
<th>StdTcpClientConnection ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual</td>
<td>~StdTcpClientConnection ()</td>
</tr>
<tr>
<td></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual bool</td>
<td>Connect (const Ptr&lt; Socket &gt; &amp;s)</td>
</tr>
<tr>
<td>bool</td>
<td>IsConnected () const</td>
</tr>
<tr>
<td>virtual void</td>
<td>Shutdown ()</td>
</tr>
<tr>
<td>const IpAddress &amp;</td>
<td>GetClientAddress () const</td>
</tr>
<tr>
<td>virtual Socket::Result</td>
<td>Send ()</td>
</tr>
<tr>
<td>virtual Socket::Result</td>
<td>Send (const Ptr&lt; IO::Stream &gt; &amp;stream)</td>
</tr>
<tr>
<td>virtual const Ptr&lt; IO::Stream &gt; &amp;</td>
<td>GetSendStream ()</td>
</tr>
<tr>
<td>virtual const Ptr&lt; IO::Stream &gt; &amp;</td>
<td>GetRecvStream ()</td>
</tr>
<tr>
<td>int</td>
<td>GetRefCount () const</td>
</tr>
<tr>
<td>void</td>
<td>AddRef ()</td>
</tr>
<tr>
<td>void</td>
<td>Release ()</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;className) const</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current reffcount

Return the current reffcount of the object.
```

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]

increment reffcount by one

Increment the reffcount of the object.
```

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]

decrement reffcount and destroy object if reffcount is zero

Decrement the reffcount and destroy object if reffcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump reffcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Net::StdTcpServer
Net::StdTcpServer Class Reference

#include <stdtcpserver.h>

Inheritance diagram for Net::StdTcpServer:
Detailed Description

A **TcpServer** opens a socket and listens for connecting TcpClients. This listen process happens in its own listener-thread and thus doesn't block the application. Each connected client is represented through a **TcpClientConnection** object which can be used by the application to communicate with a specific client.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StdTcpServer</strong>()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~StdTcpServer</strong>()</td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>void SetAddress(const IpAddress &amp;addr)</strong></td>
<td>Set address, hostname can be &quot;any&quot;, &quot;self&quot; or &quot;inetself&quot;</td>
</tr>
<tr>
<td><strong>const IpAddress &amp; GetAddress()</strong></td>
<td>Get address</td>
</tr>
<tr>
<td><strong>void SetClientConnectionClass(const Core::Rtti &amp;type)</strong></td>
<td>Set client connection class</td>
</tr>
<tr>
<td><strong>const Core::Rtti &amp; GetClientConnectionClass()</strong></td>
<td>Get client connection class</td>
</tr>
<tr>
<td><strong>bool Open()</strong></td>
<td>Open the server</td>
</tr>
<tr>
<td><strong>void Close()</strong></td>
<td>Close the server</td>
</tr>
<tr>
<td><strong>bool IsOpen()</strong></td>
<td>Return true if server is open</td>
</tr>
<tr>
<td><strong>Util::Array&lt;Ptr&lt;TcpClientConnection&gt;&gt;</strong></td>
<td>Poll clients connections for received data, call this frequently!</td>
</tr>
<tr>
<td><strong>bool Broadcast(const Ptr<a href="">IO::Stream</a> &amp;msg)</strong></td>
<td>Broadcast a message to all clients</td>
</tr>
<tr>
<td><strong>int GetRefCount()</strong></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef()</strong></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release()</strong></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>Returns <code>true</code> if the object is an instance of the given class.</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>Returns <code>true</code> if the object is an instance of the given class by string.</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>Returns <code>true</code> if the object is an instance of the given class by fourcc.</td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>Returns <code>true</code> if the object is an instance of the given class, or a derived class.</td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>Returns <code>true</code> if the object is an instance of the given class, or a derived class, by string.</td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>Returns <code>true</code> if the object is an instance of the given class, or a derived class, by fourcc.</td>
</tr>
<tr>
<td><code>GetClassName</code></td>
<td>Returns the class name.</td>
</tr>
<tr>
<td><code>GetClassFourCC</code></td>
<td>Returns the class FourCC code.</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Net::TcpClient
Net::TcpClient Class Reference

#include <tcpclient.h>

Inheritance diagram for Net::TcpClient:
Detailed Description

See StdTcpClient for details.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetBlocking (bool b)</code></td>
<td>enable/disable blocking behaviour</td>
</tr>
<tr>
<td><code>bool IsBlocking () const</code></td>
<td>get blocking behaviour</td>
</tr>
<tr>
<td><code>void SetServerAddress (const IpAddress &amp;addr)</code></td>
<td>set the server address to connect to</td>
</tr>
<tr>
<td><code>const IpAddress &amp; GetServerAddress () const</code></td>
<td>get the server address</td>
</tr>
<tr>
<td><code>Result Connect ()</code></td>
<td>establish a connection with the server</td>
</tr>
<tr>
<td><code>void Disconnect ()</code></td>
<td>disconnect from the server</td>
</tr>
<tr>
<td><code>bool IsConnected ()</code></td>
<td>return true if currently connected</td>
</tr>
<tr>
<td><code>bool Send ()</code></td>
<td>send accumulated content of send stream to server</td>
</tr>
<tr>
<td><code>const Ptr&lt; IO::Stream &gt; &amp; GetSendStream ()</code></td>
<td>access to send stream</td>
</tr>
<tr>
<td><code>bool Recv ()</code></td>
<td>receive data from server into recv stream</td>
</tr>
<tr>
<td><code>const Ptr&lt; IO::Stream &gt; &amp; GetRecvStream ()</code></td>
<td>access to recv stream</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class by fourcc</em></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class</em></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by string</em></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td><em>return true if this object is instance of given class, or a derived class, by fourcc</em></td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th></th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

StdTcpClient::Result
Net::StdTcpClient::Connect( ) [inherited]

establish a connection with the server

Establish a connection with the server. If the client is set to non-blocking at the time this method is called, it will return immediately with the result Connecting. To check if the connection is standing, just call \texttt{Connect()} again in intervals which will eventually return Success. On a blocking client, connect returns after a connection has been established, or with a time out when no connection could be established.

Reimplemented in \texttt{Net::MessageClient}.

\begin{verbatim}
void Net::StdTcpClient::Disconnect( ) [inherited]

disconnect from the server

This disconnects the current connection.

Reimplemented in \texttt{Net::MessageClient}.
\end{verbatim}

\begin{verbatim}
bool Net::StdTcpClient::IsConnected( ) [inherited]

return true if currently connected

Return true if the socket is currently connected. This will actually probe the connection using a select().
\end{verbatim}

\begin{verbatim}
int Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.
\end{verbatim}
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Net::TcpClientConnection
# Net::TcpClientConnection Class Reference

#include <tcpclientconnection.h>

Inheritance diagram for Net::TcpClientConnection:
Detailed Description

See StdTcpClientConnection for details!

(C) 2009 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool <code>Connect (const Ptr&lt; Socket &gt; &amp;s)</code></td>
<td>connect using provided socket</td>
</tr>
<tr>
<td>bool <code>IsConnected () const</code></td>
<td>get the connection status</td>
</tr>
<tr>
<td>virtual void <code>Shutdown ()</code></td>
<td>shutdown the connection</td>
</tr>
<tr>
<td>const <code>IpAddress &amp; GetClientAddress () const</code></td>
<td>get the client's ip address</td>
</tr>
<tr>
<td>virtual Socket::Result <code>Send ()</code></td>
<td>send accumulated content of send stream to server</td>
</tr>
<tr>
<td>virtual Socket::Result <code>Send (const Ptr&lt; IO::Stream &gt; &amp;stream)</code></td>
<td>directly send a stream to the server, often prevents a memory copy</td>
</tr>
<tr>
<td>virtual const <code>Ptr&lt; IO::Stream &gt; &amp; GetSendStream ()</code></td>
<td>access to send stream</td>
</tr>
<tr>
<td>virtual Socket::Result <code>Recv ()</code></td>
<td>receive data from server into recv stream</td>
</tr>
<tr>
<td>virtual const <code>Ptr&lt; IO::Stream &gt; &amp; GetRecvStream ()</code></td>
<td>access to recv stream</td>
</tr>
<tr>
<td>int <code>GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td></td>
</tr>
<tr>
<td>Return Type</td>
<td>Function</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

```c
static void DumpRefCountingLeaks() {
    dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
}
```
get the current refcount

Return the current refcount of the object.

increment refcount by one

Increment the refcount of the object.

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

get the class name

Get the class name of the object.

get the class FourCC code

Get the class FourCC of the object.

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Net::TcpMessageCodec
Net::TcpMessageCodec Class Reference

#include <tcpmessagecodec.h>
Detailed Description

Helperclass that provides function to encode and decode streams into messages.

The encoder adds header informations at the beginning of the stream, that includes the complete size of the stream. Call EncodeToMessage at the sending site.

The decoder reads sequential incoming streams and append the data to an internal buffer. It concatenates the incoming streams to a list of streams containing complete messages. Call DecodeStream at the receiving site and check for completed messages with GetMessages.

(C) 2009 Radon Labs
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TcpMessageCodec ()</strong></td>
<td>Constructor.</td>
</tr>
<tr>
<td>virtual <strong>~TcpMessageCodec ()</strong></td>
<td>Destructor.</td>
</tr>
<tr>
<td><strong>EncodeToMessage</strong> (const Ptr&lt; IO::Stream &gt; &amp;stream, const Ptr&lt; IO::Stream &gt; &amp;output)</td>
<td>Attaches header information to the stream and returns a copy with header.</td>
</tr>
<tr>
<td><strong>DecodeStream</strong> (const Ptr&lt; IO::Stream &gt; &amp;stream)</td>
<td>Decodes a given Stream. Check for HasMessages() if this completes a message.</td>
</tr>
<tr>
<td><strong>HasMessages ()</strong></td>
<td>Returns true, if there are messages in the internal message queue.</td>
</tr>
<tr>
<td><strong>Util::Array&lt; Ptr&lt; IO::Stream &gt; &gt;</strong></td>
<td><strong>Dequeuemessages ()</strong></td>
</tr>
<tr>
<td></td>
<td>Gets the list of all created messages since the last call of this function.</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Net::TcpMessageCodec::EncodeToMessage(const Ptr<IO::Stream> &stream, const Ptr<IO::Stream> &output)
```

Attaches header information to the stream and returns a copy with header.

Writes a copy of the given stream to the given output stream with header informations at the beginning.

```cpp
void Net::TcpMessageCodec::DecodeStream(const Ptr<IO::Stream> &stream)
```

Decodes a given Stream. Check for `HasMessages()` if this completes a message.

Puts a given stream to the internal buffer. This may result in new messages in the queue.

```cpp
bool Net::TcpMessageCodec::HasMessages()
```

Returns true, if there are messages in the internal message queue.

Returns true if messages are available

```cpp
Util::Array<Ptr<IO::Stream>> Net::TcpMessageCodec::DequeueMessages()
```

Gets the list of all created messages since the last call of this function.

Returns all created messages since the last call and clears message queue.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Net::TcpServer
Net::TcpServer Class Reference

#include <tcpserver.h>

Inheritance diagram for Net::TcpServer:

```
Core::RefCounted
  `-- Net::StdTcpServer
      `-- Net::TcpServer
```
Detailed Description

Front-end wrapper class for StdTcpServer, see StdTcpServer for details!

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void SetAddress() (const IpAddress &amp;addr)</td>
<td>set address, hostname can be “any”, “self” or “inetself”</td>
</tr>
<tr>
<td>const IpAddress &amp; GetAddress() const</td>
<td>get address</td>
</tr>
<tr>
<td>void SetClientConnectionClass( const Core::Rtti &amp;type)</td>
<td>set client connection class</td>
</tr>
<tr>
<td>const Core::Rtti &amp; GetClientConnectionClass()</td>
<td>get client connection class</td>
</tr>
<tr>
<td>bool Open()</td>
<td>open the server</td>
</tr>
<tr>
<td>void Close()</td>
<td>close the server</td>
</tr>
<tr>
<td>bool IsOpen() const</td>
<td>return true if server is open</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>Util::Array&lt;Ptr&lt;TcpClientConnection&gt;&gt; &amp;Recv()</td>
<td>poll clients connections for received data, call this frequently!</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application (NEBLA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount( ) const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef( ) [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release( ) [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

OSX::OSXConsoleHandler
OSX::OSXConsoleHandler Class Reference

#include <osxconsolehandler.h>

Inheritance diagram for OSX::OSXConsoleHandler:
Detailed Description

The default console handler for OSX, puts messages to stdout and stderr, reads from stdin.

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void</td>
<td><strong>Print</strong> (const <strong>Util::String</strong> &amp;s) called by console to output data</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Error</strong> (const <strong>Util::String</strong> &amp;s) called by console with serious error</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Warning</strong> (const <strong>Util::String</strong> &amp;s) called by console to output warning</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Confirm</strong> (const <strong>Util::String</strong> &amp;s) called by console to display confirmation message box</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>DebugOut</strong> (const <strong>Util::String</strong> &amp;s) called by console to output debug string</td>
</tr>
<tr>
<td>virtual bool</td>
<td><strong>HasInput</strong> () return true if input is available</td>
</tr>
<tr>
<td>virtual <strong>Util::String</strong></td>
<td><strong>GetInput</strong> () read available input</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Open</strong> () called by console when attached</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Close</strong> () called by console when removed</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsOpen</strong> () const return true if currently open</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Update</strong> () called by <strong>Console::Update</strong>()</td>
</tr>
<tr>
<td>int</td>
<td><strong>GetRefCount</strong> () const get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td><strong>AddRef</strong> () increment refcount by one</td>
</tr>
<tr>
<td>void</td>
<td><strong>Release</strong> () decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

  dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
**Member Function Documentation**

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
This method should be called as the very last before an application exits.
OSX::OSXCpu
OSX::OSXCpu Class Reference

#include <osxcpu.h>
Detailed Description

CPU related definitions for the MacOSX platform.

(C) 2010 Radon Labs GmbH
## Static Public Attributes

<table>
<thead>
<tr>
<th>static const CoreId</th>
<th>InvalidCoreId = 0xffffffff</th>
</tr>
</thead>
<tbody>
<tr>
<td>core identifiers, under Win32, we basically don't care...</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:48 2010
OSX::OSXCriticalSection
OSX::OSXCriticalSection Class Reference

#include <osxcriticalsection.h>
Detailed Description

On OSX, pthread mutexes are used for critical sections.

Todo:

: Add debugging asserts? If yes wrap with new __NEBULA3-
define

(C) 2010 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OSXCriticalSection ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~OSXCriticalSection ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void Enter () const</code></td>
<td>enter the critical section</td>
</tr>
<tr>
<td><code>void Leave () const</code></td>
<td>leave the critical section</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:48 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

OSX::OSXFileTime
#include <osxfiletime.h>
Detailed Description

Wraps file-system related timestamps on OSX.

(C) 2010 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSXFileTime ()</td>
</tr>
</tbody>
</table>

`constructor`
## Friends

<table>
<thead>
<tr>
<th>bool</th>
<th>operator== (const OSXFileTime &amp;a, const OSXFileTime &amp;b)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>operator ==</td>
</tr>
<tr>
<td>bool</td>
<td>operator!= (const OSXFileTime &amp;a, const OSXFileTime &amp;b)</td>
</tr>
<tr>
<td></td>
<td>operator !=</td>
</tr>
<tr>
<td>bool</td>
<td>operator&gt; (const OSXFileTime &amp;a, const OSXFileTime &amp;b)</td>
</tr>
<tr>
<td></td>
<td>operator &gt;</td>
</tr>
<tr>
<td>bool</td>
<td>operator&lt; (const OSXFileTime &amp;a, const OSXFileTime &amp;b)</td>
</tr>
<tr>
<td></td>
<td>operator &lt;</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

OSX::OSXFSWrapper
OSX::OSXFSWrapper Class Reference

#include <osxfswrapper.h>
Detailed Description

Internal filesystem wrapper for the OSX.

(C) 2010 Radon Labs GmbH
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OpenFile</code></td>
<td><code>static Handle OpenFile (const Util::String &amp;path, IO::Stream::AccessMode accessMode, IO::Stream::AccessPattern accessPattern)</code>&lt;br&gt;open a file</td>
</tr>
<tr>
<td><code>CloseFile</code></td>
<td><code>static void CloseFile (Handle h)</code>&lt;br&gt;close a file</td>
</tr>
<tr>
<td><code>Write</code></td>
<td><code>static void Write (Handle h, const void *buf, IO::Stream::Size numBytes)</code>&lt;br&gt;write to a file</td>
</tr>
<tr>
<td><code>Read</code></td>
<td><code>static IO::Stream::Size Read (Handle h, void *buf, IO::Stream::Size numBytes)</code>&lt;br&gt;read from a file</td>
</tr>
<tr>
<td><code>Seek</code></td>
<td><code>static void Seek (Handle h, IO::Stream::Offset offset, IO::Stream::SeekOrigin orig)</code>&lt;br&gt;seek in a file</td>
</tr>
<tr>
<td><code>Tell</code></td>
<td><code>static IO::Stream::Position Tell (Handle h)</code>&lt;br&gt;get position in file</td>
</tr>
<tr>
<td><code>Flush</code></td>
<td><code>static void Flush (Handle h)</code>&lt;br&gt;flush a file</td>
</tr>
<tr>
<td><code>Eof</code></td>
<td><code>static bool Eof (Handle h)</code>&lt;br&gt;return true if at end-of-file</td>
</tr>
<tr>
<td><code>GetFileSize</code></td>
<td><code>static IO::Stream::Size GetFileSize (Handle h)</code>&lt;br&gt;get size of a file in bytes</td>
</tr>
<tr>
<td><code>SetReadOnly</code></td>
<td><code>static void SetReadOnly (const Util::String &amp;path, bool readOnly)</code>&lt;br&gt;set read-only status of a file</td>
</tr>
<tr>
<td><code>IsReadOnly</code></td>
<td><code>static bool IsReadOnly (const Util::String &amp;path)</code>&lt;br&gt;get read-only status of a file</td>
</tr>
<tr>
<td><code>DeleteFile</code></td>
<td><code>static bool DeleteFile (const Util::String &amp;path)</code>&lt;br&gt;delete a file</td>
</tr>
<tr>
<td><code>DeleteDirectory</code></td>
<td><code>static bool DeleteDirectory (const Util::String &amp;path)</code>&lt;br&gt;delete an empty directory</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>static bool FileExists (const Util::String &amp;path)</code></td>
<td>return true if a file exists</td>
</tr>
<tr>
<td><code>static bool DirectoryExists (const Util::String &amp;path)</code></td>
<td>return true if a directory exists</td>
</tr>
<tr>
<td><code>static void SetFileWriteTime (const Util::String &amp;path, IO::FileTime &amp;fileTime)</code></td>
<td>set the write-access time stamp of a file</td>
</tr>
<tr>
<td><code>static IO::FileTime GetFileWriteTime (const Util::String &amp;path)</code></td>
<td>get the last write-access time stamp of a file</td>
</tr>
<tr>
<td><code>static bool CreateDirectory (const Util::String &amp;path)</code></td>
<td>create a directory</td>
</tr>
<tr>
<td><code>static Util::Array&lt; Util::String &gt; ListFiles (const Util::String &amp;dirPath, const Util::String &amp;pattern)</code></td>
<td>list all files in a directory</td>
</tr>
<tr>
<td><code>static Util::Array&lt; Util::String &gt; ListDirectories (const Util::String &amp;dirPath, const Util::String &amp;pattern)</code></td>
<td>list all subdirectories in a directory</td>
</tr>
<tr>
<td><code>static Util::String GetUserDirectory ()</code></td>
<td>get path to the current user's home directory (for user: standard assign)</td>
</tr>
<tr>
<td><code>static Util::String GetAppDataDirectory ()</code></td>
<td>get path to the current user's appdata directory (for appdata: standard assign)</td>
</tr>
<tr>
<td><code>static Util::String GetTempDirectory ()</code></td>
<td>get path to the current user's temp directory (for temp: standard assign)</td>
</tr>
<tr>
<td><code>static Util::String GetHomeDirectory ()</code></td>
<td>get path to the current application directory (for home: standard assign)</td>
</tr>
<tr>
<td><code>static Util::String GetBinDirectory ()</code></td>
<td>get path to the current bin directory (for bin: standard assign)</td>
</tr>
<tr>
<td><code>static Util::String GetProgramsDirectory ()</code></td>
<td>get path to the &quot;c:/program files&quot; directory</td>
</tr>
<tr>
<td><code>static bool IsDeviceName (const Util::String &amp;str)</code></td>
<td>return true when the string is a device name (e.g. &quot;C:&quot;))</td>
</tr>
</tbody>
</table>
static const char * ConvertPath (const Util::String &str)

skips the OSX: at the start of the path
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

OSX::OSXGuid
#include <osxguid.h>
Detailed Description

OSX implementation of the Util::Guid class.

(C) 2010 Radon Labs GmbH
Public Member Functions

- `void * operator new (size_t s)`
  *override new operator*

- `void operator delete (void *ptr)`
  *override delete operator*

- `OSXGuid ()`
  *constructor*

- `OSXGuid (const OSXGuid &rhs)`
  *copy constructor*

- `OSXGuid (const unsigned char *ptr, SizeT size)`
  *construct from raw binary data as returned by AsBinary()*

- `void operator= (const OSXGuid &rhs)`
  *assignment operator*

- `void operator= (const Util::String &rhs)`
  *assignment operator from string*

- `bool operator== (const OSXGuid &rhs)`
  *equality operator*

- `bool operator!= (const OSXGuid &rhs)`
  *inequality operator*

- `bool operator< (const OSXGuid &rhs)`
  *less-than operator*

- `bool operator<= (const OSXGuid &rhs)`
  *less-or-equal operator*

- `bool operator> (const OSXGuid &rhs)`
  *greater-than operator*

- `bool operator>= (const OSXGuid &rhs)`
  *greater-or-equal operator*

- `bool IsValid ()`
  *return true if the contained guid is valid (not NIL)*

- `void Generate ()`
  *generate a new guid*

- `Util::String AsString ()`
  *get as string*

- `SizeT AsBinary (const unsigned char *outPtr)`
  *const*
<table>
<thead>
<tr>
<th>IndexT</th>
<th><strong>HashCode () const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get a hash code (compatible with <code>Util::HashTable</code>)</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>OSXGuid</code> <strong>FromString</strong> (const <code>Util::String</code> &amp;str)</td>
<td>Construct from string representation</td>
</tr>
<tr>
<td>static <code>OSXGuid</code> <strong>FromBinary</strong> (const unsigned char *ptr, SizeT numBytes)</td>
<td>Construct from binary representation</td>
</tr>
<tr>
<td>static <code>SizeT</code> <strong>BinarySize</strong> ()</td>
<td>Return the size of the binary representation in bytes</td>
</tr>
</tbody>
</table>
Member Function Documentation

```
OSXGuid
OSX::OSXGuid::FromBinary( const unsigned ptr, char * numBytes
                        SizeT numBytes
                        )
[static]
```

construct from binary representation

Constructs the guid from binary data, as returned by the `AsBinary()`.

```
SizeT
OSX::OSXGuid::AsBinary( const unsigned outPtr ) const
char * outPtr
```

get pointer to binary data

This method allows read access to the raw binary data of the uuid. It returns the number of bytes in the buffer, and a pointer to the data.

```
IndexT
OSX::OSXGuid::HashCode( ) const
```

get a hash code (compatible with `Util::HashTable`)

This method returns a hash code for the uuid, compatible with `Util::HashTable`. This is simply copied from `String::HashCode`...
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

OSX::OSXHeap
#include <osxheap.h>
Detailed Description

OSX implementation of Memory::Heap. The OSX implementation uses a memory zone.

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSXHeap (const char *name, size_t initialSize=64 *1024)</td>
<td>constructor (name must be a static string!)</td>
</tr>
<tr>
<td>~OSXHeap ()</td>
<td>destructor</td>
</tr>
<tr>
<td>const char * GetName () const</td>
<td>get heap name</td>
</tr>
<tr>
<td>void * Alloc (size_t size, size_t alignment=16)</td>
<td>allocate a block of memory from the heap</td>
</tr>
<tr>
<td>void * Realloc (void *ptr, size_t newSize)</td>
<td>re-allocate a block of memory</td>
</tr>
<tr>
<td>void Free (void *ptr)</td>
<td>free a block of memory</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

```cpp
static void Setup ()

static setup method (called by Core::SysFunc::Setup)
```
Member Function Documentation

void OSX::OSXHeap::Setup() [static]

static setup method (called by Core::SysFunc::Setup)

This method must be called at the beginning of the application before any threads are spawned.
OSX::OSXInterlocked
OSX::OSXInterlocked Class Reference

#include <osxinterlocked.h>
Detailed Description

Provides simple atomic operations on shared variables.

(C) 2010 Radon Labs GmbH
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int</td>
<td>Increment</td>
<td>(int volatile &amp;var) interlocked increment</td>
</tr>
<tr>
<td>static int</td>
<td>Decrement</td>
<td>(int volatile &amp;var) interlocked decrement</td>
</tr>
<tr>
<td>static void</td>
<td>Add</td>
<td>(int volatile &amp;var, int add) interlocked add</td>
</tr>
<tr>
<td>static int</td>
<td>Exchange</td>
<td>(int volatile *dest, int value) interlocked exchange</td>
</tr>
<tr>
<td>static int</td>
<td>CompareExchange</td>
<td>(int volatile *dest, int exchange, int comparand) interlocked compare-exchange</td>
</tr>
</tbody>
</table>
OSX::OSXMemoryPool
OSX::OSXMemoryPool Class Reference

#include <osxmemorypool.h>
Detailed Description

FIXME: IMPLEMENT ME!

(C) 2010 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSXMemoryPool()</td>
<td>Constructor</td>
</tr>
<tr>
<td>~OSXMemoryPool()</td>
<td>Destructor</td>
</tr>
<tr>
<td>void Setup(Memory::HeapType heapType, uint blockSize, uint numBlocks)</td>
<td>Setup the memory pool</td>
</tr>
<tr>
<td>void * Alloc()</td>
<td>Allocate a block from the pool (NOTE: returns 0 if pool exhausted!)</td>
</tr>
<tr>
<td>void Free(void *ptr)</td>
<td>Deallocate a block from the pool</td>
</tr>
<tr>
<td>bool IsPoolBlock(void *ptr) const</td>
<td>Return true if block is owned by this pool</td>
</tr>
<tr>
<td>uint GetNumBlocks() const</td>
<td>Get number of allocated blocks in pool</td>
</tr>
<tr>
<td>uint GetBlockSize() const</td>
<td>Get block size</td>
</tr>
<tr>
<td>uint GetAlignedBlockSize() const</td>
<td>Get aligned block size</td>
</tr>
<tr>
<td>uint GetPoolSize() const</td>
<td>Get pool size</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static uint</th>
<th>ComputeAlignedBlockSize (uint blockSize)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>compute the actual block size including alignment and management data</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void OSX::OSXMemoryPool::Setup(Memory::HeapType heapType_,
                                    uint blockSize_,
                                    uint numBlocks_)

setup the memory pool

NOTE: name must be a static string!
```
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

OSX::OSXThread
#include <osxthread.h>

Inheritance diagram for OSX::OSXThread:
Detailed Description

OSX implementation of Threading::Thread. Uses the pthread API.

(C) 2010 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>thread priorities</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OSXThread ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~OSXThread ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>SetPriority</strong> (Priority p)**</td>
<td>set the thread priority</td>
</tr>
<tr>
<td><strong>Priority GetPriority () const</strong></td>
<td>get the thread priority</td>
</tr>
<tr>
<td>void <strong>SetCoreId</strong> (System::Cpu::CoreId coreId)**</td>
<td>set cpu core on which the thread should be running</td>
</tr>
<tr>
<td><strong>System::Cpu::CoreId GetCoreId () const</strong></td>
<td>get the cpu core on which the thread should be running</td>
</tr>
<tr>
<td>void <strong>SetStackSize</strong> (SizeT s)**</td>
<td>set stack size in bytes (default is 4 KByte)</td>
</tr>
<tr>
<td><strong>SizeT GetStackSize () const</strong></td>
<td>get stack size</td>
</tr>
<tr>
<td>void <strong>SetName</strong> (const Util::String &amp;n)**</td>
<td>set thread name</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetName () const</strong></td>
<td>get thread name</td>
</tr>
<tr>
<td>void <strong>Start ()</strong></td>
<td>start executing the thread code, returns when thread has actually started</td>
</tr>
<tr>
<td>void <strong>Stop ()</strong></td>
<td>request threading code to stop, returns when thread has actually finished</td>
</tr>
<tr>
<td>bool <strong>IsRunning () const</strong></td>
<td>return true if thread has been started</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void YieldThread ()</code></td>
<td><em>yield the thread (gives up current time slice)</em></td>
</tr>
<tr>
<td><code>static Threading::ThreadId GetMyThreadId ()</code></td>
<td><em>get the thread ID of this thread</em></td>
</tr>
<tr>
<td><code>static void DumpRefCountingLeaks ()</code></td>
<td><em>dump refcounting leaks, call at end of application</em></td>
</tr>
<tr>
<td></td>
<td>(<em>NEBULA3_DEBUG builds only!</em>)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>EmitWakeupSignal</strong> ()</td>
<td>override this method if your thread loop needs a wakeup call before stopping</td>
</tr>
<tr>
<td>virtual void <strong>DoWork</strong> ()</td>
<td>this method runs in the thread context</td>
</tr>
<tr>
<td>bool <strong>ThreadStopRequested</strong> () const</td>
<td>check if stop is requested, call from <strong>DoWork()</strong> to see if the thread proc should quit</td>
</tr>
</tbody>
</table>
set thread name

Set the thread's name.

get thread name

Get the thread's name. This is the vanilla method which returns the name member.

request threading code to stop, returns when thread has actually finished

This stops the thread by signalling the stopRequestEvent and waits for the thread to actually quit. If the thread code runs in a loop it should use the IsStopRequested() method to see if the thread object wants it to shutdown. If so DoWork() should simply return.

return true if thread has been started

Returns true if the thread is currently running.

yield the thread (gives up current time slice)
Give up time slice.

Threading::ThreadId
OSX::OSXThread::GetMyThreadId() [static]

get the thread ID of this thread

Static method which returns the ThreadId of this thread.

void
OSX::OSXThread::EmitWakeupSignal() [protected, virtual]

override this method if your thread loop needs a wakeup call before stopping

This method is called by Thread::Stop() after setting the stopRequest event and before waiting for the thread to stop. If your thread runs a loop and waits for jobs it may need an extra wakeup signal to stop waiting and check for the ThreadStopRequested() event. In this case, override this method and signal your event object.

void
OSX::OSXThread::DoWork() [protected, virtual]

this method runs in the thread context

This method should be derived in a Thread subclass and contains the actual code which is run in the thread. The method must not call C-Lib functions under Win32. To terminate the thread, just return from this function. If DoWork() runs in an infinite loop, call ThreadStopRequested() to check whether the Thread object wants the thread code to quit.

bool
OSX::OSXThread::ThreadStopRequested() const [inline, protected]

check if stop is requested, call from DoWork() to see if the thread proc should quit

If the derived DoWork() method is running in a loop it must regularly check if the process wants the thread to terminate by calling
ThreadStopRequested() and simply return if the result is true. This will cause the thread to shut down.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

OSX::OSXThreadld
OSX::OSXThreadId Class Reference

#include <osxthreadid.h>
Detailed Description

A thread id uniquely identifies a thread within the process.

(C) 2010 Radon Labs GmbH
OSX::OSXThreadLocalPtr
OSX::OSXThreadLocalPtr< TYPE >  
Class Template Reference

#include <osxthreadlocalptr.h>
Detailed Description

template<typename TYPE>
class OSX::OSXThreadLocalPtr< TYPE >

GCC doesn't implement the __thread modifier on OSX. Instead we use pthread keys to emulate the behaviour.

Todo:
- Performance? Would it actually be better to allocate one pointer lookup-table and associate this with a single thread-local key as like on the Wii? At the moment every Ptr has its own key.

Todo:
- If this object will be used for anything else then singleton, it might make sense to change the interface to look like a normal C-pointer.

(C) 2010 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSXThreadLocalPtr ()</td>
<td>default constructor</td>
</tr>
<tr>
<td>~OSXThreadLocalPtr ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void set (TYPE *p)</td>
<td>set content</td>
</tr>
<tr>
<td>TYPE * get () const</td>
<td>get content</td>
</tr>
<tr>
<td>bool isvalid () const</td>
<td>test if content is valid</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:48 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

OSX::SysFunc
#include <osxsysfunc.h>
Detailed Description

Lowest-level functions for OSX platform.

(C) 2010 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Setup()</code></td>
<td>setup lowlevel Nebula3 runtime (called before anything else)</td>
</tr>
<tr>
<td><code>Exit(int exitCode)</code></td>
<td>cleanly exit the process</td>
</tr>
<tr>
<td><code>Error(const char *error)</code></td>
<td>display an error message</td>
</tr>
<tr>
<td><code>MessageBox(const char *msg)</code></td>
<td>display a message which must be confirmed by the user</td>
</tr>
<tr>
<td><code>DebugOut(const char *msg)</code></td>
<td>print a message on the debug console</td>
</tr>
<tr>
<td><code>Sleep(double sec)</code></td>
<td>sleep for a specified amount of seconds</td>
</tr>
<tr>
<td><code>GetSystemInfo()</code></td>
<td>get system information</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](https://www.doxygen.org/) at Fri Mar 26 15:21:48 2010
Particles::EmitterAttrs
Particles::EmitterAttrs Class Reference

#include <emitterattrs.h>
Detailed Description

A container for particle emitter attributes.

(C) 2008 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>Enum</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>FloatAttr</td>
<td>scalar attributes</td>
</tr>
<tr>
<td>enum</td>
<td>BoolAttr</td>
<td>boolean attributes</td>
</tr>
<tr>
<td>enum</td>
<td>IntAttr</td>
<td>integer attributes</td>
</tr>
<tr>
<td>enum</td>
<td>EnvelopeAttr</td>
<td>scalar envelope attributes</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EmitterAttrs ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>void <strong>SetFloat</strong> (FloatAttr key, float value)</td>
<td>set float attribute</td>
</tr>
<tr>
<td>float <strong>GetFloat</strong> (FloatAttr key) const</td>
<td>get float attribute</td>
</tr>
<tr>
<td>void <strong>SetBool</strong> (BoolAttr key, bool value)</td>
<td>set bool attribute</td>
</tr>
<tr>
<td>bool <strong>GetBool</strong> (BoolAttr key) const</td>
<td>get bool attribute</td>
</tr>
<tr>
<td>void <strong>SetInt</strong> (IntAttr key, int value)</td>
<td>set int attribute</td>
</tr>
<tr>
<td>int <strong>GetInt</strong> (IntAttr key) const</td>
<td>get int attribute</td>
</tr>
<tr>
<td>void <strong>SetEnvelope</strong> (EnvelopeAttr key, const EnvelopeCurve &amp;value)</td>
<td>set envelope attribute</td>
</tr>
<tr>
<td>const EnvelopeCurve &amp; <strong>GetEnvelope</strong> (EnvelopeAttr key) const</td>
<td>get envelope attribute</td>
</tr>
</tbody>
</table>
Particles::EmitterMesh
Particles::EmitterMesh Class Reference

#include <emittermesh.h>
Detailed Description

An emitter mesh holds positions and normals for particle emission. The actual format of a vertex in the emitter mesh is:

float4 position; float4 normal; float4 tangent;

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EmitterMesh ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~EmitterMesh ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void Setup (const Ptr&lt; CoreGraphics::Mesh &gt; &amp;mesh, IndexT primGroupIndex)</td>
<td>setup the emitter mesh</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>discard the emitter mesh</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>return true if object has been setup</td>
</tr>
<tr>
<td>const EmitterPoint &amp; GetEmitterPoint (IndexT key) const</td>
<td>get emitter point</td>
</tr>
</tbody>
</table>
Particles::EnvelopeCurve
Particles::EnvelopeCurve Class Reference

#include <envelopecurve.h>
Detailed Description

An Attack/Sustain/Release envelope curve class with optional sine/cosine modulation. Used for animated particle emitter attributes.

(C) 2008 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>ModFunc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>modulation enumerator</td>
</tr>
</tbody>
</table>
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EnvelopeCurve ()</strong></td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td>void <strong>Setup</strong></td>
<td>(float val0, float val1, float val2, float val3, float keyPos0, float keyPos1, float freq, float amp, <strong>ModFunc</strong> mod) set parameters</td>
</tr>
<tr>
<td>float <strong>Sample</strong></td>
<td>(float t) const sample at specific time (0..1)</td>
</tr>
<tr>
<td>void <strong>PreSample</strong></td>
<td>(float *sampleBuffer, SizeT numSamples, SizeT sampleStride) const sample from t=0 to t=1 into array of values</td>
</tr>
<tr>
<td>float <strong>GetMaxValue</strong></td>
<td>() const get the max of val0, val1, val2, val3</td>
</tr>
</tbody>
</table>
Member Function Documentation

float
Particles::EnvelopeCurve::Sample ( float t ) const

sample at specific time (0..1)

NOTE: Sampling a single value is relatively expensive. Consider pre-sampling into a lookup table!

void
Particles::EnvelopeCurve::PreSample ( float * sampleBuffer,
                                       SizeT numSamples,
                                       SizeT sampleStride
                                 ) const

sample from t=0 to t=1 into array of values

This samples N values from t=0 to t=1 into an array. The array can then be used as a lookup table.
Particles::EnvelopeSampleBuffer
Particles::EnvelopeSampleBuffer Class Reference

#include <envelopesamplebuffer.h>
Detailed Description

A lookup table for pre-sampled envelope curves.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EnvelopeSampleBuffer ()</strong></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td><strong>~EnvelopeSampleBuffer ()</strong></td>
<td><em>destructor</em></td>
</tr>
<tr>
<td><strong>void Setup</strong> (const EmitterAttrs &amp;emitterAttrs, SizeT numSamples)**</td>
<td>setup the sample buffer</td>
</tr>
<tr>
<td><strong>void Discard ()</strong></td>
<td>discard the sample buffer</td>
</tr>
<tr>
<td><strong>bool IsValid () const</strong></td>
<td>return true if object has been setup</td>
</tr>
<tr>
<td><strong>IndexT AsSampleIndex (float t) const</strong></td>
<td>convert t-value (0.0 to 1.0) into a lookup index</td>
</tr>
<tr>
<td><strong>float * LookupSamples</strong> (IndexT sampleIndex) const**</td>
<td>get pointer to samples, index into array by <code>EmitterAttrs::EnvelopeAttr</code></td>
</tr>
<tr>
<td><strong>SizeT GetNumSamples () const</strong></td>
<td>get the number of samples per attribute</td>
</tr>
<tr>
<td><strong>const float * GetSampleBuffer () const</strong></td>
<td>get the sample buffer</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](http://www.doxygen.org/) at Fri Mar 26 15:21:48 2010
Particles::Particle
#include <particle.h>
Detailed Description

The particle structure holds the current state of a single particle and common data for particle-job and nebula3 particle system

!! NOTE: this header is also included from job particlejob.cc, so only !! job-compliant headers can be included here

(C) 2008 Radon Labs GmbH
Particles::ParticleRenderer
Particles::ParticleRenderer Class Reference

#include <particlerenderer.h>

Inheritance diagram for Particles::ParticleRenderer:
Detailed Description

Platform-wrapper for particle rendering.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ParticleRenderer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~ParticleRenderer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>Setup ()</strong></td>
<td>Setup the particle renderer</td>
</tr>
<tr>
<td>virtual void <strong>Discard ()</strong></td>
<td>Discard the particle renderer</td>
</tr>
<tr>
<td>virtual void <strong>BeginAttach ()</strong></td>
<td>Begin attaching visible particle systems</td>
</tr>
<tr>
<td>virtual void <strong>EndAttach ()</strong></td>
<td>End attaching visible particle systems</td>
</tr>
<tr>
<td><strong>GetCurParticleVertexIndex ()</strong> const</td>
<td>Get the current vertex index</td>
</tr>
<tr>
<td><strong>AddCurParticleVertexIndex (IndexT add)</strong></td>
<td>Add particle vertex index</td>
</tr>
<tr>
<td><strong>GetCurVertexPtr ()</strong></td>
<td>Get the current vertex pointer</td>
</tr>
<tr>
<td>void *<em>SetCurVertexPtr (void <em>ptr)</em></em></td>
<td>Set the current vertex pointer</td>
</tr>
<tr>
<td>const <strong>GetParticleVertexBuffer ()</strong> const</td>
<td>Get particle vertex buffer</td>
</tr>
<tr>
<td>const <strong>GetCornerVertexBuffer ()</strong> const</td>
<td>Get the corner vertex buffer</td>
</tr>
<tr>
<td>const <strong>GetCornerIndexBuffer ()</strong> const</td>
<td>Get the corner index buffer</td>
</tr>
<tr>
<td><strong>GetPrimitiveGroup ()</strong></td>
<td>Get the primitive group</td>
</tr>
<tr>
<td>const <strong>GetVertexLayout ()</strong> const</td>
<td>Get the vertex layout</td>
</tr>
<tr>
<td><strong>IsValid ()</strong></td>
<td>IsValid</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td><code>AddVisibleParticleSystem</code></td>
<td>attach a visible particle system instance</td>
</tr>
<tr>
<td><code>AddVisibleParticleSystem</code></td>
<td>(const `Ptr&lt; Particles::ParticleSystemInstance &gt; &amp;particleSystemInstance) attach a visible particle system instance</td>
</tr>
<tr>
<td><code>IsInAttach</code></td>
<td>is renderer in attach?</td>
</tr>
<tr>
<td><code>RenderParticleSystem</code></td>
<td>render particles of previously attached particle system</td>
</tr>
<tr>
<td><code>RenderParticleSystem</code></td>
<td>(const `Ptr&lt; Particles::ParticleSystemInstance &gt; &amp;particleSystemInstance) render particles of previously attached particle system</td>
</tr>
<tr>
<td><code>GetRefCount</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>(const `Rtti &amp;rtti) const return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>(const `Util::String &amp;className) const return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>(const `Util::FourCC &amp;classFourCC) const return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>(const `Rtti &amp;rtti) const return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>(const `Util::String &amp;className) const return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

- `return true if particle renderer has been setup`
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
void Direct3D9::D3D9ParticleRenderer::BeginAttach() [virtual, inherited]

begin attaching visible particle systems

This method is called once per frame before visible particle systems are attached.

Reimplemented from Base::ParticleRendererBase.

void Direct3D9::D3D9ParticleRenderer::EndAttach() [virtual, inherited]

finish attaching visible particle systems

This method is called once per frame after visible particle systems are attached.

Reimplemented from Base::ParticleRendererBase.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Particles::ParticleRenderInfo
Particles::ParticleRenderInfo Class Reference

#include <particlerenderinfo.h>
Detailed Description

ParticleRenderInfo objects are returned by the ParticleRenderer singleton when a visible particle system is attached. The caller needs to store this object and needs to hand it back to the ParticleRenderer when actually rendering of the particle system needs to happen.

(C) 2008 Radon Labs GmbH
Particles::ParticleServer
Particles::ParticleServer Class Reference

#include <particleserver.h>

Inheritance diagram for Particles::ParticleServer:
Detailed Description

Setup the particles subsystem.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ParticleServer ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~ParticleServer ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>void <code>Open ()</code></td>
<td>setup particle subsystem</td>
</tr>
<tr>
<td>void <code>Close ()</code></td>
<td>discard particle subsystem</td>
</tr>
<tr>
<td>bool <code>IsOpen () const</code></td>
<td>return true if particle subsystem has been setup</td>
</tr>
<tr>
<td>int <code>GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>Util::FourCC</code></th>
<th><code>GetClassFourCC()</code> const</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>get the class FourCC code</code></td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Particles::ParticleSystem
Particles::ParticleSystem Class Reference

#include <particlesystem.h>

Inheritance diagram for Particles::ParticleSystem:

```
Core::RefCounted

Particles::ParticleSystem
```
Detailed Description

A ParticleSystem object holds the shared attributes for all its ParticleSystemInstances.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ParticleSystem</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~ParticleSystem</strong> ()</td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>Setup</strong> (const Ptr&lt; CoreGraphics::Mesh &gt; &amp;emitterMesh, IndexT primGroupIndex, const EmitterAttrs &amp;emitterAttrs)</td>
<td>Set up the particle system object</td>
</tr>
<tr>
<td><strong>Discard</strong> ()</td>
<td>Discard the particle system object</td>
</tr>
<tr>
<td><strong>IsValid</strong> () const</td>
<td>Return true if object has been setup</td>
</tr>
<tr>
<td>const EmitterMesh &amp; <strong>GetEmitterMesh</strong> () const</td>
<td>Get pointer to emitter mesh</td>
</tr>
<tr>
<td>const EmitterAttrs &amp; <strong>GetEmitterAttrs</strong> () const</td>
<td>Get access to particle emitter attributes</td>
</tr>
<tr>
<td>const EnvelopeSampleBuffer &amp; <strong>GetEnvelopeSampleBuffer</strong> () const</td>
<td>Get access to pre-sampled envelope curve buffer</td>
</tr>
<tr>
<td><strong>GetMaxNumParticles</strong> () const</td>
<td>Get the maximum number of particles alive</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef</strong> ()</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><strong>Release</strong> ()</td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::FourCC ...) const</td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>&amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetClassName</strong> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong> <strong>GetClassFourCC</strong> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>ModulateStepTime (float val)</th>
</tr>
</thead>
<tbody>
<tr>
<td>modulate stepTime by a factor, needed for time effects</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
<td></td>
</tr>
</tbody>
</table>
## Static Public Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static const <code>Timing::Time</code></td>
<td><strong>DefaultStepTime</strong></td>
<td>the default particle system update step time</td>
</tr>
<tr>
<td>static <code>Timing::Time</code></td>
<td><strong>StepTime</strong></td>
<td>the particle system update step time</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
This method should be called as the very last before an application exits.
Particles::ParticleSystemNode
#include <particlesystemnode.h>

Inheritance diagram for Particles::ParticleSystemNode:
Detailed Description

The **ParticleSystemNode** wraps a **ParticleSystem** object into a ModelNode for rendering.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ParticleSystemNode()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~ParticleSystemNode()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual Ptr&lt; Models::ModelNodeInstance &gt;</code> CreateNodeInstance()` const</td>
<td>Create a model node instance</td>
</tr>
<tr>
<td><code>virtual void</code> OnAttachToModel(const Ptr&lt; Models::Model &gt; &amp;model)`</td>
<td>Called when attached to model node</td>
</tr>
<tr>
<td><code>virtual void</code> OnRemoveFromModel()</td>
<td>Called when removed from model node</td>
</tr>
<tr>
<td><code>virtual void</code> LoadResources()`</td>
<td>Called when resources should be loaded</td>
</tr>
<tr>
<td><code>virtual void</code> UnloadResources()`</td>
<td>Called when resources should be unloaded</td>
</tr>
<tr>
<td><code>virtual Resources::Resource::State</code> GetResourceState()` const</td>
<td>Get overall state of contained resources (Init, Loaded, Pending, Failed, Cancelled)</td>
</tr>
<tr>
<td><code>virtual void</code> OnResourcesLoaded()`</td>
<td>Called once when all pending resource have loaded</td>
</tr>
<tr>
<td><code>virtual bool</code> ParseDataTag(const Util::FourCC &amp;fourCC, const Ptr&lt; IO::BinaryReader &gt; &amp;reader)`</td>
<td>Parse data tag (called by loader code)</td>
</tr>
<tr>
<td><code>const Ptr&lt; ParticleSystem &gt; &amp;</code> GetParticleSystem()` const</td>
<td>Get the owned ParticleSystem object</td>
</tr>
<tr>
<td><code>void</code> SetEmitterMeshResourceId(const Resources::ResourceId &amp;resId)`</td>
<td>Set resource id of emitter mesh</td>
</tr>
<tr>
<td><code>const Resources::ResourceId &amp;</code> GetEmitterMeshResourceId()`</td>
<td>Get emitter mesh resource id</td>
</tr>
<tr>
<td><code>void</code> SetPrimitiveGroupIndex(IndexT i)`</td>
<td>Set the primitive group index in the emitter mesh</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetPrimitiveGroupIndex</code></td>
<td>get the primitive group index in the emitter mesh</td>
</tr>
<tr>
<td><code>SetEmitterAttrs</code></td>
<td>set emitter attributes</td>
</tr>
<tr>
<td><code>GetEmitterAttrs</code></td>
<td>get emitter attributes</td>
</tr>
<tr>
<td><code>ApplySharedState</code></td>
<td>apply state shared by all my ModelNodeInstances</td>
</tr>
<tr>
<td><code>SetShader</code></td>
<td>set shader resource id</td>
</tr>
<tr>
<td><code>GetShader</code></td>
<td>get shader resource id</td>
</tr>
<tr>
<td><code>GetShaderInstance</code></td>
<td>get pointer to contained shader instance</td>
</tr>
<tr>
<td><code>AddShaderParam</code></td>
<td>add optional shader parameter, must be called before LoadResources</td>
</tr>
<tr>
<td><code>GetShaderParameter</code></td>
<td>get shaderparams</td>
</tr>
<tr>
<td><code>GetPosition</code></td>
<td>set position</td>
</tr>
<tr>
<td><code>GetRotation</code></td>
<td>set rotate quaternion</td>
</tr>
<tr>
<td><code>GetScale</code></td>
<td>set scale</td>
</tr>
<tr>
<td><code>SetPosition</code></td>
<td>set position</td>
</tr>
<tr>
<td><code>SetRotation</code></td>
<td>set rotate quaternion</td>
</tr>
<tr>
<td><code>SetScale</code></td>
<td>set scale</td>
</tr>
</tbody>
</table>
void SetRotatePivot (const Math::point &p)
    set rotate pivot

const Math::point & GetRotatePivot () const
    get rotate pivot

void SetScalePivot (const Math::point &)
    set scale pivot

const Math::point & GetScalePivot () const
    get scale pivot

bool IsInViewSpace () const
    is transformnode in viewspace

void SetInViewSpace (bool b)
    set transformnode in viewspace

float GetMinDistance () const
    get MinDistance

void SetMinDistance (float val)
    set MinDistance

float GetMaxDistance () const
    get MaxDistance

void SetMaxDistance (float val)
    set MaxDistance

bool LodDistancesUsed () const
    are lod distances used

bool GetLockedToViewer () const
    get LockedToViewer

void SetLockedToViewer (bool val)
    set LockedToViewer

bool CheckLodDistance (float distToViewer) const
    helper method to check whether the distance within lod distances

Ptr< ModelNodeInstance > CreateNodeInstanceHierarchy (Ptr< ModelInstance > &modelInst)
    recursively create model node instance and model node instances

virtual void BeginParseDataTags ()
    begin parsing data tags
virtual void EndParseDataTags ()

bool IsAttachedToModel () const
return true if currently attached to a Model

const Ptr< Model > & GetModel () const
get model this node is attached to

void SetResourceStreamingLevelOfDetail (float factor)
sets the resourceStreamingLevelOfDetail but
the given value is bigger than the current one
(reseted on frame-start)

void ResetScreenSpaceStats ()
resets all screen space stats e.g. size

void SetBoundingBox (const Math::bbox &b)
set bounding box

const Math::bbox & GetBoundingBox () const
get bounding box of model node

void SetName (const Util::StringAtom &)
set model node name

const Util::StringAtom & GetName () const
get model node name

void SetType (ModelNodeType::Code)
set ModelNodeType

ModelNodeType::Code GetType () const
get the ModelNodeType

void SetParent (const Ptr< ModelNode > &p)
set parent node

const Ptr< ModelNode > & GetParent () const
get parent node

bool HasParent () const
return true if node has a parent

void AddChild (const Ptr< ModelNode > &c)
add a child node

const Util::Array< Ptr< ModelNode > > & GetChildren () const
get child nodes

bool HasChild (const Util::StringAtom &name) const
return true if a direct child exists by name

const Ptr<ModelNode> & LookupChild (const Util::StringAtom &name) const
get pointer to direct child by name

void AddVisibleNodeInstance (IndexT frameIndex, const Ptr<ModelNodeInstance> &nodeInst)
called by model node instance on NotifyVisible()

const Util::Array< Ptr<ModelNodeInstance> > & GetVisibleModelNodeInstances (ModelNodeType::Code t) const
get visible model node instances

bool HasStringAttr (const Util::StringAtom &attrId) const
has string attr

const Util::StringAtom & GetStringAttr (const Util::StringAtom &attrId) const
get string attr

void SetStringAttr (const Util::StringAtom &attrId, const Util::StringAtom &value)
add string attribute

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC
<table>
<thead>
<tr>
<th>bool</th>
<th>&amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dumps refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void</strong></td>
<td><strong>SetupManagedTextureVariable</strong> (const Resources::ResourceId &amp;texResId, const Ptr<a href="">CoreGraphics::ShaderVariable</a> &amp;var)</td>
</tr>
<tr>
<td></td>
<td>setup a new managed texture variable</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>UpdateManagedTextureVariables</strong> (IndexT frameIndex)</td>
</tr>
<tr>
<td></td>
<td>update managed texture variables</td>
</tr>
<tr>
<td><strong>virtual</strong></td>
<td><strong>RecurseCreateNodeInstanceHierarchy</strong> (const Ptr&lt;ModelInstance&gt; &amp;modelInst, const Ptr&lt;ModelNodeInstance&gt; &amp;parentNodeInst)</td>
</tr>
<tr>
<td></td>
<td>recursively create node instance hierarchy</td>
</tr>
</tbody>
</table>
## Protected Attributes

<table>
<thead>
<tr>
<th>float</th>
<th><code>resourceStreamingLevelOfDetail</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>factor between 0.0 (close) and 1.0 (far away) describing the distance to camera (used for decision of max needed mipMap)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::StateNode::AddShaderParam
    (const Util::String &paramName,
     const Util::Variant &paramValue &)
    [inherited]
```

add optional shader parameter, must be called before LoadResources

Manual shader parameters must be added before LoadResources is called, because on LoadResources all shader params are validated.

```cpp
void Models::StateNode::SetupManagedTextureVariable
    (const Resources::ResourceId &texResId,
     const Ptr<CoreGraphics::ShaderVariable> &var &)
    [protected, inherited]
```

setup a new managed texture variable

Create a new managed texture resource and bind it to the provided shader variable.

```cpp
void Models::StateNode::UpdateManagedTextureVariables
    (IndexT frameIndex ) [protected, inherited]
```

update managed texture variables

This method transfers texture from our managed texture objects into their associated shader variable. This is necessary since the actual texture of a managed texture may change from frame to frame because of resource management.

```cpp
Ptr<ModelNodeInstance>
Models::ModelNode::CreateNodeInstanceHierarchy
    (const ModellInstance modellInst ) [inherited]
```
recursively create model node instance and child model node instances

Create the node instance hierarchy.

```c++
void Models::ModelNode::BeginParseDataTags() [virtual, inherited]
```

begin parsing data tags

Begin parsing data tags. This method is called by `StreamModelLoader` before `ParseDataTag()` is called for the first time.

```c++
void Models::ModelNode::EndParseDataTags() [virtual, inherited]
```

finish parsing data tags

End parsing data tags. This method is called by `StreamModelLoader` after the last `ParseDataTag()` is called.

```c++
void Models::ModelNode::ResetScreenSpaceStats() [inline, inherited]
```

resets all screen space stats e.g. size

Reset resourceStreamingLevelOfDetail to -1.0 as we are able to recognize invisible items this way. (visible items will overwrite this value with a value >= 0.0)

```c++
Ptr< ModelNodeInstance > Models::ModelNode::RecurseCreateNodeInstanceHierarchy(const Ptr< ModelInstance > & modellinst,
const Ptr< ModelNodeInstance > & parentNodeInst )
```

recursively create node instance hierarchy

Recursively create node instances and attach them to the provided
model instance. Returns a pointer to the root node instance.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Particles::ParticleSystemNodeInstance
Particles::ParticleSystemNodeInstance
Class Reference

#include <particlesystemnodeinstance.h>

Inheritance diagram for Particles::ParticleSystemNodeInstance:

- Core::RefCounted
- Models::ModelNodeInstance
- Models::TransformNodeInstance
- Models::StateNodeInstance
- Particles::ParticleSystemNodeInstance
Detailed Description

Wraps a ParticleSystemInstance into a ModelNodeInstance.

(C) 2008 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ParticleSystemNodeInstance</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>virtual ~ParticleSystemNodeInstance</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual void OnRenderBefore (IndexT frameIndex, Timing::Time time)</code></td>
<td>called from <code>ModelEntity::OnRenderBefore</code></td>
</tr>
<tr>
<td><code>virtual void OnVisibilityResolve (IndexT resolveIndex, float distanceToViewer)</code></td>
<td>called during visibility resolve</td>
</tr>
<tr>
<td><code>virtual void ApplyState ()</code></td>
<td>apply per-instance state prior to rendering</td>
</tr>
<tr>
<td><code>virtual void Render ()</code></td>
<td>perform rendering</td>
</tr>
<tr>
<td><code>const Ptr&lt; ParticleSystemInstance &gt; &amp; GetParticleSystemInstance</code></td>
<td>get the node's particle system instance</td>
</tr>
<tr>
<td><code>Ptr&lt; CoreGraphics::ShaderVariableInstance &gt;</code></td>
<td>CreateShaderVariableInstance (CoreGraphics::ShaderVariable::Semantic &amp;semantic)</td>
</tr>
<tr>
<td><code>bool HasShaderVariableInstance (CoreGraphics::ShaderVariable::Semantic &amp;semantic) const</code></td>
<td>return true if a shader variable has been instanciated</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::ShaderVariableInstance &gt; &amp; GetShaderVariableInstance (CoreGraphics::ShaderVariable::Semantic &amp;semantic) const</code></td>
<td>get a shader variable instance</td>
</tr>
<tr>
<td><code>void SetPosition (const Math::point &amp;)</code></td>
<td>set position</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetPosition () const</code></td>
<td>get position</td>
</tr>
<tr>
<td><code>void SetRotate (const Math::quaternion &amp;)</code></td>
<td>set rotate quaternion</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const Math::quaternion &amp; GetRotate()</code></td>
<td>get rotate quaternion</td>
</tr>
<tr>
<td><code>void SetScale (const Math::vector &amp;)</code></td>
<td>set scale</td>
</tr>
<tr>
<td><code>const Math::vector &amp; GetScale()</code></td>
<td>get scale</td>
</tr>
<tr>
<td><code>void SetRotatePivot (const Math::point &amp;)</code></td>
<td>set rotate pivot</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetRotatePivot()</code></td>
<td>get rotate pivot</td>
</tr>
<tr>
<td><code>void SetScalePivot (const Math::point &amp;)</code></td>
<td>set scale pivot</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetScalePivot()</code></td>
<td>get scale pivot</td>
</tr>
<tr>
<td><code>void SetOffsetMatrix (const Math::matrix44 &amp;)</code></td>
<td>set optional offset matrix</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetOffsetMatrix()</code></td>
<td>get optional offset matrix</td>
</tr>
<tr>
<td><code>bool IsInViewSpace ()</code></td>
<td>is transformnode in viewspace</td>
</tr>
<tr>
<td><code>void SetInViewSpace (bool b)</code></td>
<td>set transformnode in viewspace</td>
</tr>
<tr>
<td><code>bool GetLockedToViewer ()</code></td>
<td>get LockedToViewer</td>
</tr>
<tr>
<td><code>void SetLockedToViewer (bool val)</code></td>
<td>set LockedToViewer</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetLocalTransform()</code></td>
<td>get resulting local transform matrix in local parent space</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetModelTransform()</code></td>
<td>get model space transform (valid after Update())</td>
</tr>
<tr>
<td><code>virtual void DiscardHierarchy ()</code></td>
<td>discard the model node instance and all of its children</td>
</tr>
<tr>
<td><code>bool IsValid ()</code></td>
<td>return true if the model node instance is valid</td>
</tr>
<tr>
<td><code>virtual void OnNotifyCullingVisible (IndexT)</code></td>
<td></td>
</tr>
</tbody>
</table>
const Util::StringAtom & GetName () const
get model node name

bool HasParent () const
return true if node has a parent

const Ptr< ModelNodeInstance > & GetParent () const
get parent node

const Util::Array< Ptr< ModelNodeInstance > > & GetChildren () const
get child nodes

bool HasChild (const Util::StringAtom &name) const
return true if a direct child exists by name

const Ptr< ModelNodeInstance > & LookupChild (const Util::StringAtom &name) const
get pointer to direct child by name

Ptr< ModelNodeInstance > LookupPath (const Util::String
get model node instance by hierarchy path

const Ptr< ModelInstance > & GetModelInstance () const
get the ModelInstance we are attached to

const Ptr< ModelNode > & GetModelNode () const
get the ModelNode we're associated with

void SetVisible (bool b, Timing::Time recursive=true)
set the node instance's visibility

bool IsVisible () const
return true if node instance is set to visible

IndexT GetModelNodeInstanceIndex
get model node instance index for current frame

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti)
return true if this object is instance of given class
<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class or</td>
</tr>
<tr>
<td></td>
<td>derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of</td>
</tr>
<tr>
<td></td>
<td>given class or derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of</td>
</tr>
<tr>
<td></td>
<td>given class or derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>Util::String</strong> &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>DumpRefCountingLeaks()</code></td>
<td><code>dump refcounting leaks, call at end of application</code> (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual void Setup (const Ptr&lt; Models::ModelInstance &gt; &amp;inst, const Ptr&lt; Models::ModelNode &gt; &amp;node, const Ptr&lt; Models::ModelNodeInstance &gt; &amp;parentNodeInst)</code></td>
<td>called when attached to <code>ModelInstance</code></td>
</tr>
<tr>
<td><code>virtual void Discard ()</code></td>
<td>called when removed from <code>ModelInstance</code></td>
</tr>
<tr>
<td><code>virtual void RenderDebug ()</code></td>
<td>render node specific debug shape</td>
</tr>
<tr>
<td><code>virtual void OnShow (Timing::Time time)</code></td>
<td>called when the node becomes visible with current time</td>
</tr>
<tr>
<td><code>virtual void OnHide (Timing::Time time)</code></td>
<td>called when the node becomes invisible</td>
</tr>
<tr>
<td><code>void SetParent (const Ptr&lt; ModelNodeInstance &gt; &amp;p)</code></td>
<td>set parent node</td>
</tr>
<tr>
<td><code>void AddChild (const Ptr&lt; ModelNodeInstance &gt; &amp;c)</code></td>
<td>add a child node</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Models::ModelNodeInstance::DiscardHierarchy() [virtual, inherited]
discard the model node instance and all of its children
Discards this model node instance and all of its children recursively.

bool Models::ModelNodeInstance::IsVisible() const [inherited]
return true if node instance is set to visible
FIXME: The recursion in this is method makes it slow, especially in deep hierarchies. Also it can't be inlined.

int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
get the class name

Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Particles::ParticleSystemRenderer
Particles::ParticleSystemRenderer
Class Reference

#include <particlesysteminstance.h>
Detailed Description

Platform-wrapper for particle-system-instance rendering.

(C) 2008 Radon Labs GmbH
Particles::ParticleSystemState
Particles::ParticleSystemState Class Reference

#include <particlesystemstate.h>
Detailed Description

State bits of a particle system instance.

(C) 2008 Radon Labs GmbH
Public Types

enum Bits
  particle system state bits
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields
PFeedbackLoop Class Reference

#include <pfeedbackloop.h>
Detailed Description

A P feedback loop (proportional feedback loop) is a simple object which moves a system’s current state towards a goal, using the resulting error (difference between goal and state as feedback on the next run.

If you need to implement motion controllers, camera controllers, etc... then the feedback loop is your friend.

See Game Developer Mag issue June/July 2004.

(C) 2007 RadonLabs GmbH
PhysicsFeature::ActorPhysicsProperty
PhysicsFeature::ActorPhysicsProperty
Class Reference

#include <actorphysicsproperty.h>

Inheritance diagram for PhysicsFeature::ActorPhysicsProperty:
Detailed Description

**ActorPhysicsProperty** adds "actor physics" to an entity. This is mainly a capsule which is always kept upright and responds "immediately" to move messages. Attach the **ActorPhysicsProperty** instead of a **PhysicsProperty** to an entity.

**ActorPhysicsProperty** implements the following messages:

- MoveGoto
- MoveFollow
- MoveDirection
- MoveStop
- MoveTurn
- MoveRotate

(C) 2008 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>CallbackType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>callback types</em></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ActorPhysicsProperty</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~ActorPhysicsProperty</strong> ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>SetupCallbacks</strong> ()</td>
<td>setup callbacks for this property, call by entity in <strong>OnActivate</strong>()</td>
</tr>
<tr>
<td>virtual void <strong>SetupDefaultAttributes</strong> ()</td>
<td>setup default entity attributes</td>
</tr>
<tr>
<td>virtual void <strong>OnActivate</strong> ()</td>
<td>called from Entity::ActivateProperties()</td>
</tr>
<tr>
<td>virtual void <strong>OnDeactivate</strong> ()</td>
<td>called from Entity::DeactivateProperties()</td>
</tr>
<tr>
<td>virtual void <strong>OnMoveBefore</strong> ()</td>
<td>called before movement has happened</td>
</tr>
<tr>
<td>virtual void <strong>OnMoveAfter</strong> ()</td>
<td>called after movement has happened</td>
</tr>
<tr>
<td>virtual void <strong>OnRenderDebug</strong> ()</td>
<td>called on debug visualization</td>
</tr>
<tr>
<td>virtual void <strong>OnLoseActivity</strong> ()</td>
<td>called on losing activity (no longer triggered)</td>
</tr>
<tr>
<td>virtual <strong>Ptr&lt; Physics::PhysicsEntity &gt;</strong></td>
<td><strong>GetPhysicsEntity</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get a pointer to the physics entity</td>
</tr>
<tr>
<td>virtual void <strong>SetupAcceptedMessages</strong> ()</td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>virtual void <strong>HandleMessage</strong> (const <strong>Ptr&lt;</strong></td>
<td>**Messaging::Message &gt; &amp;msg)</td>
</tr>
<tr>
<td></td>
<td>handle a single message</td>
</tr>
<tr>
<td><strong>SetEnabled</strong> (bool enabled)</td>
<td>enable/disable physics</td>
</tr>
<tr>
<td><strong>IsEnabled</strong> () const</td>
<td>is physics enabled</td>
</tr>
<tr>
<td><strong>GetEntity</strong> () const</td>
<td>get entity</td>
</tr>
<tr>
<td><strong>GetEntity</strong> () const</td>
<td>get entity</td>
</tr>
</tbody>
</table>
bool HasEntity() const
return true if entity pointer is valid

bool IsActive() const
return true if property is currently active

virtual void OnLoad()
called from within Entity::Load() after attributes are loaded

virtual void OnStart()
called from within Entity::OnStart() after OnLoad when the complete world exist

virtual void OnSave()
called from within Entity::Save() before attributes are saved back to database

virtual void OnBeginFrame()
called on begin of frame

virtual void OnRender()
called before rendering happens

virtual void OnGainActivity()
called when game debug visualization is on

void AttachHandler(const Ptr< Handler >& h)
attach a message handler to the port

void RemoveHandler(const Ptr< Handler >& h)
remove a message handler from the port

void RemoveAllHandlers()
remove all message handler from the port

SizeT GetNumHandlers() const
return number of handlers attached to the port

const Ptr< Handler >& GetHandlerAtIndex(IndexT i) const
get a message handler by index

virtual void Send(const Ptr< Message >& msg)
send a message to the port

const Util::Array< const Id * >& GetAcceptedMessages() const
get the array of accepted messages (sorted)

bool AcceptsMessage(const Id &msgId) const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void EnablePhysics()</td>
<td>enable and activate the physics</td>
</tr>
<tr>
<td>virtual void DisablePhysics()</td>
<td>disable and cleanup the physics</td>
</tr>
<tr>
<td>void Stop()</td>
<td>immediately stop the entity</td>
</tr>
<tr>
<td>void SendStop()</td>
<td>send a synchronous MoveStop message to self</td>
</tr>
<tr>
<td>void HandleMoveDirection()</td>
<td>handle a MoveDirection message</td>
</tr>
<tr>
<td>void HandleMoveGoto()</td>
<td>handle a MoveGoto message</td>
</tr>
<tr>
<td>void HandleSetTransform()</td>
<td>handle a SetTransform message</td>
</tr>
<tr>
<td>void HandleMoveTurn()</td>
<td>handle a MoveTurn message</td>
</tr>
<tr>
<td>void HandleMoveRotate()</td>
<td>handle a MoveRotate message</td>
</tr>
<tr>
<td>void HandleMoveFollow()</td>
<td>handle a MoveFollow message</td>
</tr>
<tr>
<td>bool IsGotoActive() const</td>
<td>return true if Goto is currently active</td>
</tr>
<tr>
<td>void ContinueGoto()</td>
<td>continue a Goto</td>
</tr>
<tr>
<td>void ContinueFollow()</td>
<td>continue a Follow</td>
</tr>
</tbody>
</table>
void SkipSegments ()
    skip segments

void AutoEvade (Math::vector &targetVec)
    perform auto evade

virtual Ptr< Physics::PhysicsEntity > CreatePhysicsEntity () const
    create physics entity

void ApplyImpulseAtPos (const Math::vector &impulse, const Math::vector &pos, bool multByMass=false)
    apply a global impulse vector at the next time step at a global position

void SetEntity (const Ptr< Entity > &v)
    Set entity, this is attached to, to `v'.

void ClearEntity ()
    Remove entity.

void RegisterMessage (const Id &msgId)
    register a single accepted message
## Static Protected Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static const float</td>
<td><strong>AutoEvadeProbeAboveGround</strong></td>
<td>dist of auto evade probe to entity's pos</td>
</tr>
<tr>
<td>static const float</td>
<td><strong>AutoEvadeProbeRadius</strong></td>
<td>radius of auto evade probe</td>
</tr>
</tbody>
</table>
Member Function Documentation

```c++
void PhysicsFeature::ActorPhysicsProperty::OnMoveBefore() [virtual]
called before movement has happened

The **OnMoveBefore()** method handles all pending messages, and other per-frame-stuff that must happen before the physics subsystem is triggered.

Reimplemented from **Game::Property**.

```c++
void PhysicsFeature::ActorPhysicsProperty::OnMoveAfter() [virtual]
called after movement has happened

The **OnMoveAfter()** method transfers the current physics entity transform to the game entity.

Reimplemented from **PhysicsFeature::PhysicsProperty**.

```c++
void PhysicsFeature::ActorPhysicsProperty::OnRenderDebug() [virtual]
called on debug visualization

Render a debug visualization of the current 3d navigation path.

Reimplemented from **Game::Property**.

```c++
void PhysicsFeature::ActorPhysicsProperty::EnablePhysics() [protected, virtual]
enable and activate the physics

Creates a Physics::CharEntity instead of a normal Physics::PhysicsEntity.
Reimplemented from `PhysicsFeature::PhysicsProperty`.

```cpp
void
PhysicsFeature::ActorPhysicsProperty::Stop() [protected]
```

immediately stop the entity

Immediately stop the entity.

26-Jan-06 floh bugfix: also cancelled MoveFollow 14-Feb-06 nico bugfix: now really cancelled MoveFollow ;)

```cpp
void
PhysicsFeature::ActorPhysicsProperty::SendStop() [protected]
```

send a synchronous MoveStop message to self

This simply sends a synchronous stop message to myself. This uses a message so that everybody else who might be interested in the information that I have stopped can listen to the message.

```cpp
void
PhysicsFeature::ActorPhysicsProperty::HandleMoveDirection(BaseGameFeature::MoveDirection *)
```

handle a MoveDirection message

Handle a MoveDirection message.

```cpp
void
PhysicsFeature::ActorPhysicsProperty::HandleMoveGoto(BaseGameFeature::MoveGoto * msg)
```

handle a MoveGoto message

Handle a MoveGoto message.

```cpp
void
PhysicsFeature::ActorPhysicsProperty::HandleSetTransform(BaseGameFeature::SetTransform *)
```

handle a SetTransform message

Handle a SetTransform message.
handle a MoveTurn message
Handle a MoveTurn message.

handle a MoveRotate message
Handle a MoveTurn message.

handle a MoveFollow message
Handle a MoveFollow message.

continue a Goto
Continue the current Goto action.

continue a Follow
Continue the current Follow action.

perform auto evade
Function was changed(11.07.07). Now AutoEvade()will be called by HandleMoveDirection() to avoid replacement of non-static physic objekts.
void PhysicsFeature::PhysicsProperty::ApplyImpulseAtPos(const Math::vector impulse, &
const Math::vector pos, &
bool multByMass = false)

[protected, inherited]

apply a global impulse vector at the next time step at a global position

Apply an impulse vector at a position in the global coordinate frame.

void Game::Property::OnLoad() [virtual, inherited]

called from within Entity::Load() after attributes are loaded

This method is called from within Game::Entity::Load() after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void Game::Property::OnStart() [virtual, inherited]

called from within Entity::OnStart() after OnLoad when the complete world exist

This method is called from within Game::Entity::OnStart(). This is the moment when the world is complete and the entity can establish connections to other entities.

Reimplemented in GraphicsFeature::CameraProperty.
Game::Property::OnSave

called from within Entity::Save() before attributes are saved back to database

This method is called from within Game::Entity::Save() before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void Game::Property::OnBeginFrame() [virtual, inherited]
called on begin of frame

This method is called from Game::Entity::OnBeginFrame() on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void Game::Property::OnRender() [virtual, inherited]
called before rendering happens

This method is called from Game::Entity::OnRender() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before rendering happens.

Reimplemented in GraphicsFeature::CameraProperty, GraphicsFeature::ChaseCameraProperty, and GraphicsFeature::MayaCameraProperty.
called when game debug visualization is on

This method is called from `Game::Entity::OnRenderDebug()` on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

attach a message handler to the port

Attach a message handler to the port.

remove a message handler from the port

Remove a message handler from the port.

send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.
get the current refcount

Return the current refcount of the object.

```cpp
void
Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void
Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
PhysicsFeature::EnvironmentCollideProperty
PhysicsFeature::EnvironmentCollideProp
Class Reference

#include <environmentcollideproperty.h>

Inheritance diagram for PhysicsFeature::EnvironmentCollideProperty:
**Detailed Description**

This property adds pieces of static collide geometry to the game world. It is very similar to the class EnvironmentGraphicsProperty, but instead of graphics it handles collision. All static collide geometry in a level will usually be added to one EnvironmentCollideProperty, which in turn lives in a single game entity which represent the environment graphics and collision. That way the game entity pool isn't flooded with hundreds of game entities which would end up doing nothing because they just represent static geometry. Instead, everything static about the level is put into a single entity.

NOTE: usually you don't need to care about this class, it's used by the level loader which automatically collects all environment objects into a single game entity.

(C) 2008 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>CallbackType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>callback types</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EnvironmentCollideProperty()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~EnvironmentCollideProperty()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>OnDeactivate()</code></td>
<td>called from <code>Entity::DeactivateProperties()</code></td>
</tr>
<tr>
<td><code>AddShapes</code></td>
<td>add a collide shape group</td>
</tr>
<tr>
<td><code>DeleteShapes</code></td>
<td>delete shapes associated with given id</td>
</tr>
<tr>
<td><code>HasShapes</code></td>
<td>return true if a shape group of the given id exists</td>
</tr>
<tr>
<td><code>GetShapes</code></td>
<td>get shape group by id</td>
</tr>
<tr>
<td><code>GetLocalMatrices</code></td>
<td>get world matrix associated with id</td>
</tr>
<tr>
<td><code>GetEntity</code></td>
<td>get entity this property is attached to</td>
</tr>
<tr>
<td><code>HasEntity</code></td>
<td>return true if entity pointer is valid</td>
</tr>
<tr>
<td><code>SetupDefaultAttributes()</code></td>
<td>setup the property's attributes to their default state</td>
</tr>
<tr>
<td><code>SetupCallbacks()</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void OnActivate()</td>
<td>called from Entity::ActivateProperties()</td>
</tr>
<tr>
<td>bool IsActive()</td>
<td>return true if property is currently active</td>
</tr>
<tr>
<td>virtual void OnLoad()</td>
<td>called from within Entity::Load() after attributes are loaded</td>
</tr>
<tr>
<td>virtual void OnStart()</td>
<td>called from within Entity::OnStart() after OnLoad when the complete world exist</td>
</tr>
<tr>
<td>virtual void OnSave()</td>
<td>called from within Entity::Save() before attributes are saved back to database</td>
</tr>
<tr>
<td>virtual void OnBeginFrame()</td>
<td>called on begin of frame</td>
</tr>
<tr>
<td>virtual void OnMoveBefore()</td>
<td>called before movement happens</td>
</tr>
<tr>
<td>virtual void OnMoveAfter()</td>
<td>called after movement has happened</td>
</tr>
<tr>
<td>virtual void OnRender()</td>
<td>called before rendering happens</td>
</tr>
<tr>
<td>virtual void OnRenderDebug()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void OnLoseActivity()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void OnGainActivity()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void HandleMessage(const Ptr<a href="">Messaging::Message</a> &amp;msg)</td>
<td>handle a single message</td>
</tr>
<tr>
<td>virtual void SetupAcceptedMessages()</td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>void AttachHandler(const Ptr&lt;Handler&gt; &amp;h)</td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void RemoveHandler (const Ptr&lt; Handler &gt; &amp;h)</code></td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td><code>void RemoveAllHandlers ()</code></td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td><code>SizeT GetNumHandlers () const</code></td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td><code>const Ptr&lt; Handler &gt; &amp; GetHandlerAtIndex (IndexT i) const</code></td>
<td>get a message handler by index</td>
</tr>
<tr>
<td><code>virtual void Send (const Ptr&lt; Message &gt; &amp;msg)</code></td>
<td>send a message to the port</td>
</tr>
<tr>
<td><code>const Util::Array&lt; const Id * &gt; &amp; GetAcceptedMessages () const</code></td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td><code>bool AcceptsMessage (const Id &amp;msgId) const</code></td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA (const Util::String &amp;rttiName) const</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><code>GetClassName () const</code></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><code>GetClassFourCC () const</code></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetEntity</strong> (const <strong>Ptr</strong>&lt; <strong>Entity</strong> &gt; &amp;v)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Set entity, this is attached to, to <code>v</code>.</em>*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>ClearEntity</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Remove entity.</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>RegisterMessage</strong> (const <strong>Id</strong> &amp;msgId)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>register a single accepted message</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Game::Property::SetupDefaultAttributes() [virtual, inherited]
```

setup the property's attributes to their default state

If a property adds attributes to an entity, override this method to setup their default state. This method is called before the entity is even initialized from the database. After this method, entity attributes may be overwritten from the database, and after that from a stream.


```cpp
void Game::Property::SetupCallbacks() [virtual, inherited]
```

setup callbacks for this property, call by entity in `OnActivate()`

Tells the entity what per-frame callback methods should be called for this property. The method is called after `SetupDefaultAttributes()` by the entity, and the property is expected to call the `Entity::RegisterPropertyCallback()` once for every callback method (`OnBeginFrame()`, `OnMoveBefore()`, ...) that should be called per-frame.

Reimplemented in `GraphicsFeature::CameraProperty`, `GraphicsFeature::GraphicsProperty`,
GraphicsFeature::MayaCameraProperty, PhysicsFeature::ActorPhysicsProperty, PhysicsFeature::PhysicsProperty, PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

```cpp
void Game::Property::OnActivate( ) [virtual, inherited]
called from Entity::ActivateProperties()

This method is called by Game::Entity::ActivateProperties(). Use this method for one-time initializations of the property.

```

```cpp
void Game::Property::OnLoad( ) [virtual, inherited]
called from within Entity::Load() after attributes are loaded

This method is called from within Game::Entity::Load() after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateGraphicsProperty.
```

```cpp
void Game::Property::OnStart( ) [virtual, inherited]
called from within Entity::OnStart() after OnLoad when the complete
world exist

This method is called from within `Game::Entity::OnStart()`. This is the moment when the world is complete and the entity can establish connections to other entities.

Reimplemented in `GraphicsFeature::CameraProperty`.

```cpp
void Game::Property::OnSave() [virtual, inherited]
```
called from within `Entity::Save()` before attributes are saved back to database

This method is called from within `Game::Entity::Save()` before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in `PhysicsFeature::TriggerProperty`, and `StateObjectFeature::StateProperty`.

```cpp
void Game::Property::OnBeginFrame() [virtual, inherited]
```
called on begin of frame

This method is called from `Game::Entity::OnBeginFrame()` on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in `PhysicsFeature::TriggerProperty`, and `StateObjectFeature::StateProperty`.

```cpp
void Game::Property::OnMoveBefore() [virtual, inherited]
```
called before movement happens
This method is called from `Game::Entity::OnMoveBefore()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before the physics subsystem is triggered.

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`.

```cpp
void Game::Property::OnMoveAfter() [virtual, inherited]
```
called after movement has happened

This method is called from `Game::Entity::OnMoveAfter()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`, and `PhysicsFeature::PhysicsProperty`.

```cpp
void Game::Property::OnRender() [virtual, inherited]
```
called before rendering happens

This method is called from `Game::Entity::OnRender()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before rendering happens.

Reimplemented in `GraphicsFeature::CameraProperty`, `GraphicsFeature::ChaseCameraProperty`, and `GraphicsFeature::MayaCameraProperty`.

```cpp
void Game::Property::OnRenderDebug() [virtual, inherited]
```
called when game debug visualization is on

This method is called from `Game::Entity::OnRenderDebug()` on all properties attached to an entity in the order of attachment. It's meant
for debug issues. It will be called when debug mode is enabled.

Reimplemented in GraphicsFeature::GraphicsProperty, PhysicsFeature::ActorPhysicsProperty, and PhysicsFeature::TriggerProperty.

```cpp
void Game::Property::OnLoseActivity() { } [virtual, inherited]
```
called when game debug visualization is on

This method is called from Game::Entity::OnLoseActivity() on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

Reimplemented in PhysicsFeature::ActorPhysicsProperty.

```cpp
void Game::Property::OnGainActivity() { } [virtual, inherited]
```
called when game debug visualization is on

This method is called from Game::Entity::OnRenderDebug() on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

```cpp
void Messaging::Port::AttachHandler(const Ptr<Handler> & h) { } [inherited]
```
attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler(const Ptr<Handler> h) { } [inherited]
```
remove a message handler from the port

Remove a message handler from the port.

```cpp
void Messaging::Port::Send (const Ptr<
    Message> &msg ) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the
HandleMessage() method of all attached handlers. If the message
has been handled by at least one of the handlers, the Handled() flag
of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

get the class name
Get the class name of the object.

`Util::FourCC`

Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

`void`

Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:49 2010
PhysicsFeature::MouseGripperProperty
PhysicsFeature::MouseGripperProperty
Class Reference

#include <mousegripperproperty.h>
Detailed Description

The MouseGripperProperty allows picking dynamic physics object and throw it around.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MouseGripperProperty ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~MouseGripperProperty ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>SetupCallbacks ()</strong></td>
<td>Setup callbacks for this property</td>
</tr>
<tr>
<td>virtual void <strong>SetupDefaultAttributes ()</strong></td>
<td>Setup default entity attributes</td>
</tr>
<tr>
<td>virtual void <strong>OnActivate ()</strong></td>
<td>Called from Entity::ActivateProperties()</td>
</tr>
<tr>
<td>virtual void <strong>OnDeactivate ()</strong></td>
<td>Called from Entity::DeactivateProperties()</td>
</tr>
<tr>
<td>virtual void <strong>OnBeginFrame ()</strong></td>
<td>Called after movement has happened</td>
</tr>
<tr>
<td>virtual void <strong>OnMoveBefore ()</strong></td>
<td>Called before movement has happened</td>
</tr>
<tr>
<td>virtual void <strong>OnMoveAfter ()</strong></td>
<td>Called after movement has happened</td>
</tr>
<tr>
<td>virtual void <strong>OnRenderDebug ()</strong></td>
<td>On render debug</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><strong>HandleLeftMouseBtnDown</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>handle left mouse btn</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>HandleLeftMouseBtnUp</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>handle left mouse btn</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

void PhysicsFeature::MouseGripperProperty::OnActivate() [virtual]
called from Entity::ActivateProperties()
Called when property is attached to a game entity. This will create and
setup the required physics entities.

void PhysicsFeature::MouseGripperProperty::OnDeactivate() [virtual]
called from Entity::DeactivateProperties()
Called when property is going to be removed from its game entity. This
will release the physics entity owned by the game entity.

void PhysicsFeature::MouseGripperProperty::OnBeginFrame() [virtual]
called after movement has happened
Handle general input.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

PhysicsFeature::PhysicsProperty
#include <physicsproperty.h>

Inheritance diagram for PhysicsFeature::PhysicsProperty:
Detailed Description

A physics property adds basic physical behaviour to a game entity. The default behaviour is that of a passive physics object which will just passively roll and bounce around. Implement more advanced behaviour in subclasses.

The physics property maintains the attributes:

Attr::Transform Attr::Velocity

(C) 2008 Radon Labs GmbH
Public Types

enum **CallbackType**
callback types
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PhysicsProperty ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~PhysicsProperty ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <strong>SetupCallbacks ()</strong></td>
<td>setup callbacks for this property</td>
</tr>
<tr>
<td>virtual void <strong>SetupDefaultAttributes ()</strong></td>
<td>setup default entity attributes</td>
</tr>
<tr>
<td>virtual void <strong>OnActivate ()</strong></td>
<td>called from <code>Entity::ActivateProperties()</code></td>
</tr>
<tr>
<td>virtual void <strong>OnDeactivate ()</strong></td>
<td>called from <code>Entity::DeactivateProperties()</code></td>
</tr>
<tr>
<td>virtual void <strong>SetupAcceptedMessages ()</strong></td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>virtual void <strong>HandleMessage</strong> (const <strong>Ptr</strong>&lt; <strong>Messaging::Message</strong>&gt; &amp;msg)</td>
<td>handle a single message</td>
</tr>
<tr>
<td>virtual void <strong>OnMoveAfter ()</strong></td>
<td>called after movement has happened</td>
</tr>
<tr>
<td>virtual <strong>Ptr</strong>&lt; <strong>Physics::PhysicsEntity</strong>&gt; <strong>GetPhysicsEntity ()</strong> const</td>
<td>get a pointer to the physics entity</td>
</tr>
<tr>
<td>void <strong>SetEnabled</strong> (bool enabled)</td>
<td>enable/disable physics</td>
</tr>
<tr>
<td>bool <strong>IsEnabled ()</strong> const</td>
<td>is physics enabled</td>
</tr>
<tr>
<td>const <strong>Ptr</strong>&lt; <strong>Entity</strong> &gt; &amp; <strong>GetEntity ()</strong> const</td>
<td>get entity this property is attached to</td>
</tr>
<tr>
<td>bool <strong>HasEntity ()</strong> const</td>
<td>return true if entity pointer is valid</td>
</tr>
<tr>
<td>bool <strong>IsActive ()</strong> const</td>
<td>return true if property is currently active</td>
</tr>
<tr>
<td>virtual void <strong>OnLoad ()</strong></td>
<td>called from within <code>Entity::Load()</code> after attributes are</td>
</tr>
</tbody>
</table>
virtual void **OnStart** ()  
called from within **Entity::OnStart()** after **OnLoad** when the complete world exist

virtual void **OnSave** ()  
called from within **Entity::Save()** before attributes are saved back to database

virtual void **OnBeginFrame** ()  
called on begin of frame

virtual void **OnMoveBefore** ()  
called before movement happens

virtual void **OnRender** ()  
called before rendering happens

virtual void **OnRenderDebug** ()  
called when game debug visualization is on

virtual void **OnLoseActivity** ()  
called when game debug visualization is on

virtual void **OnGainActivity** ()  
called when game debug visualization is on

void **AttachHandler** (const **Ptr**< **Handler** > &h)  
attach a message handler to the port

void **RemoveHandler** (const **Ptr**< **Handler** > &h)  
remove a message handler from the port

void **RemoveAllHandlers** ()  
remove all message handler from the port

**SizeT** **GetNumHandlers** () const  
return number of handlers attached to the port

const **Ptr**< **Handler** > & **GetHandlerAtIndex** (IndexT i) const  
get a message handler by index

virtual void **Send** (const **Ptr**< **Message** > &msg)  
send a message to the port

const **Util::Array**< const Id * > & **GetAcceptedMessages** () const  
get the array of accepted messages (sorted)

**bool** **AcceptsMessage** (const Id &msgId) const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetRefCount () const</code></td>
<td>Return true if port accepts this msg</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>string GetClassName () const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>Get the class FourCC code</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
**Protected Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>EnablePhysics</strong> ()</td>
<td>enable and activate the physics, overload in subclass</td>
</tr>
<tr>
<td>virtual void <strong>DisablePhysics</strong> ()</td>
<td>disable and cleanup the physics, overload in subclass</td>
</tr>
<tr>
<td>void <strong>ApplyImpulseAtPos</strong> (const Math::vector &amp;impulse, const Math::vector &amp;pos, bool multByMass=false)</td>
<td>apply a global impulse vector at the next time step at a global position</td>
</tr>
<tr>
<td>void <strong>SetEntity</strong> (const Ptr&lt; Entity &gt; &amp;v)</td>
<td>Set entity, this is attached to, to <code>v</code>.</td>
</tr>
<tr>
<td>void <strong>ClearEntity</strong> ()</td>
<td>Remove entity.</td>
</tr>
<tr>
<td>void <strong>RegisterMessage</strong> (const Id &amp;msgId)</td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

void
PhysicsFeature::PhysicsProperty::OnActivate() [virtual]
called from Entity::ActivateProperties()
Called when property is attached to a game entity. This will create and setup the required physics entities.
Reimplemented from Game::Property.
Reimplemented in PhysicsFeature::ActorPhysicsProperty.

void
PhysicsFeature::PhysicsProperty::OnDeactivate() [virtual]
called from Entity::DeactivateProperties()
Called when property is going to be removed from its game entity. This will release the physics entity owned by the game entity.
Reimplemented from Game::Property.
Reimplemented in PhysicsFeature::ActorPhysicsProperty.

void
PhysicsFeature::PhysicsProperty::OnMoveAfter() [virtual]
called after movement has happened
Called after the physics subsystem has been triggered. This will transfer the physics entity's new transform back into the game entity.
Reimplemented from Game::Property.
Reimplemented in PhysicsFeature::ActorPhysicsProperty.

Ptr<Physics::PhysicsEntity>
PhysicsFeature::PhysicsProperty::GetPhysicsEntity() const [virtual]
get a pointer to the physics entity

Get pointer to physics entity. Note that this method may return 0!

Reimplemented in **PhysicsFeature::ActorPhysicsProperty**.

```cpp
void
PhysicsFeature::PhysicsProperty::ApplyImpulseAtPos( const Math::vector &impulse, 
&
const Math::vector pos, 
&
bool multByMass = false )
[protected]
```

apply a global impulse vector at the next time step at a global position

Apply an impulse vector at a position in the global coordinate frame.

```cpp
void
Game::Property::OnLoad( ) [virtual, inherited]
```

called from within **Entity::Load()** after attributes are loaded

This method is called from within Game::Entity::Load() after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in **PhysicsFeature::TriggerProperty**, and **StateObjectFeature::StateProperty**.

```cpp
void
Game::Property::OnStart( ) [virtual, inherited]
```

called from within **Entity::OnStart()** after OnLoad when the complete world exist

This method is called from within Game::Entity::OnStart(). This is the
moment when the world is complete and the entity can establish connections to other entities.

Reimplemented in GraphicsFeature::CameraProperty.

```cpp
void
Game::Property::OnSave( ) [virtual, inherited]
```
called from within Entity::Save() before attributes are saved back to database

This method is called from within Game::Entity::Save() before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

```cpp
void
Game::Property::OnBeginFrame( ) [virtual, inherited]
```
called on begin of frame

This method is called from Game::Entity::OnBeginFrame() on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

```cpp
void
Game::Property::OnMoveBefore( ) [virtual, inherited]
```
called before movement happens

This method is called from Game::Entity::OnMoveBefore() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before the physics
subsystem is triggered.

Reimplemented in **PhysicsFeature::ActorPhysicsProperty**.

```cpp
void Game::Property::OnRender() [virtual, inherited]
```
called before rendering happens

This method is called from **Game::Entity::OnRender()** on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before rendering happens.

Reimplemented in **GraphicsFeature::CameraProperty**, **GraphicsFeature::ChaseCameraProperty**, and **GraphicsFeature::MayaCameraProperty**.

```cpp
void Game::Property::OnRenderDebug() [virtual, inherited]
```
called when game debug visualization is on

This method is called from **Game::Entity::OnRenderDebug()** on all properties attached to an entity in the order of attachment. It's meant for debug issues. It will be called when debug mode is enabled.

Reimplemented in **GraphicsFeature::GraphicsProperty**, **PhysicsFeature::ActorPhysicsProperty**, and **PhysicsFeature::TriggerProperty**.

```cpp
void Game::Property::OnLoseActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from **Game::Entity::OnLoseActivity()** on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active camera).
Reimplemented in **PhysicsFeature::ActorPhysicsProperty**.

```cpp
void
Game::Property::OnGainActivity() [virtual, inherited]
```

called when game debug visualization is on

This method is called from **Game::Entity::OnRenderDebug()** on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

```cpp
void
Messaging::Port::AttachHandler(const Ptr<Handler> &h) [inherited]
```

attach a message handler to the port

**Attach a message handler to the port.**

```cpp
void
Messaging::Port::RemoveHandler(const Ptr<Handler> &h) [inherited]
```

remove a message handler from the port

**Remove a message handler from the port.**

```cpp
void
Messaging::Port::Send(const Ptr<Message> &msg) [virtual, inherited]
```

send a message to the port

**Send a message to the port.** This will immediately call the **HandleMessage()** method of all attached handlers. If the message has been handled by at least one of the handlers, the **Handled()** flag of the message will be set to true.
int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:49 2010
PhysicsFeature::TriggerProperty
#include <triggerproperty.h>

Inheritance diagram for PhysicsFeature::TriggerProperty:
Detailed Description

Property for a defined trigger area, a trigger checks if the accepted entity is in its area and executes actions (on this entities, if action requested it)

(C) 2008 Radon Labs GmbH
Public Types

enum CallbackType

callback types
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TriggerProperty ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~TriggerProperty ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>SetupCallbacks ()</strong></td>
<td>Setup callbacks for this property</td>
</tr>
<tr>
<td><strong>SetupDefaultAttributes ()</strong></td>
<td>Setup default entity attributes</td>
</tr>
<tr>
<td><strong>OnActivate ()</strong></td>
<td>Called from Entity::ActivateProperties()</td>
</tr>
<tr>
<td><strong>OnDeactivate ()</strong></td>
<td>Called from Entity::DeactivateProperties()</td>
</tr>
<tr>
<td><strong>OnLoad ()</strong></td>
<td>Called from within Entity::Load() after attributes are loaded</td>
</tr>
<tr>
<td><strong>OnSave ()</strong></td>
<td>Called when all is saved</td>
</tr>
<tr>
<td><strong>OnBeginFrame ()</strong></td>
<td>Called on begin of frame</td>
</tr>
<tr>
<td><strong>SetupAcceptedMessages ()</strong></td>
<td>Setup accepted messages</td>
</tr>
<tr>
<td><strong>HandleMessage (const Ptr&lt;Message&gt; &amp;msg)</strong></td>
<td>Handle a single message</td>
</tr>
<tr>
<td><strong>OnRenderDebug ()</strong></td>
<td>Render a debug visualization of the trigger if needed.</td>
</tr>
<tr>
<td><strong>RenderDebugVisualization ()</strong></td>
<td>Render a debug visualization of the trigger.</td>
</tr>
<tr>
<td><strong>GetEntity () const</strong></td>
<td>Get entity this property is attached to</td>
</tr>
<tr>
<td><strong>HasEntity () const</strong></td>
<td>Return true if entity pointer is valid</td>
</tr>
<tr>
<td><strong>IsActive () const</strong></td>
<td></td>
</tr>
</tbody>
</table>

**const Ptr<Entity> &**
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <strong>OnStart</strong> ()</td>
<td><code>called from within Entity::OnStart() after OnLoad when the complete world exist</code></td>
</tr>
<tr>
<td>virtual void <strong>OnMoveBefore</strong> ()</td>
<td><code>called before movement happens</code></td>
</tr>
<tr>
<td>virtual void <strong>OnMoveAfter</strong> ()</td>
<td><code>called after movement has happened</code></td>
</tr>
<tr>
<td>virtual void <strong>OnRender</strong> ()</td>
<td><code>called before rendering happens</code></td>
</tr>
<tr>
<td>virtual void <strong>OnLoseActivity</strong> ()</td>
<td><code>called when game debug visualization is on</code></td>
</tr>
<tr>
<td>virtual void <strong>OnGainActivity</strong> ()</td>
<td><code>called when game debug visualization is on</code></td>
</tr>
<tr>
<td>void <strong>AttachHandler</strong> (const Ptr&lt; Handler &gt; &amp;h)</td>
<td><code>attach a message handler to the port</code></td>
</tr>
<tr>
<td>void <strong>RemoveHandler</strong> (const Ptr&lt; Handler &gt; &amp;h)</td>
<td><code>remove a message handler from the port</code></td>
</tr>
<tr>
<td>void <strong>RemoveAllHandlers</strong> ()</td>
<td><code>remove all message handler from the port</code></td>
</tr>
<tr>
<td>SizeT <strong>GetNumHandlers</strong> () const</td>
<td><code>return number of handlers attached to the port</code></td>
</tr>
<tr>
<td>const Ptr&lt; Handler &gt; &amp; <strong>GetHandlerAtIndex</strong> (IndexT i) const</td>
<td><code>get a message handler by index</code></td>
</tr>
<tr>
<td>virtual void <strong>Send</strong> (const Ptr&lt; Message &gt; &amp;msg)</td>
<td><code>send a message to the port</code></td>
</tr>
<tr>
<td>const Util::Array&lt; const Id * &gt; &amp; <strong>GetAcceptedMessages</strong> () const</td>
<td><code>get the array of accepted messages (sorted)</code></td>
</tr>
<tr>
<td>bool <strong>AcceptsMessage</strong> (const Id &amp;msgId) const</td>
<td><code>return true if port accepts this msg</code></td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td><code>get the current refcount</code></td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td><code>increment refcount by one</code></td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td><code>decrement refcount and destroy object if refcount is zero</code></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td><code>return true if this object is instance of given class</code></td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td><code>return true if this object is instance of given class by string</code></td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td><code>return true if this object is instance of given class by fourcc</code></td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td><code>return true if this object is instance of given class, or a derived class</code></td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td><code>return true if this object is instance of given class, or a derived class, by string</code></td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td><code>return true if this object is instance of given class, or a derived class, by fourcc</code></td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td><code>get the class name</code></td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td><code>get the class FourCC code</code></td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual void TriggerAction()</code></td>
<td>This message is called every <code>Attr::TriggerPhase</code> from timestamp</td>
</tr>
<tr>
<td><code>Math::point GetTriggerPosition()</code></td>
<td>Get position of the trigger</td>
</tr>
<tr>
<td><code>Math::vector GetTriggerScale()</code></td>
<td>Get scaling of the trigger</td>
</tr>
<tr>
<td><code>const Math::matrix44 GetTriggerTransform()</code></td>
<td>Get Trigger transform</td>
</tr>
<tr>
<td><code>void FilterEntities()</code></td>
<td>Filter current entities in <code>Trigger</code>, this will refresh <code>enterdEntities</code>,</td>
</tr>
<tr>
<td></td>
<td><code>insideEntities</code> and <code>leftEntities</code> arrays</td>
</tr>
<tr>
<td><code>void SetCurrentEntitiesInTrigger()</code></td>
<td>Will update the <code>currEntitiesInTrigger</code></td>
</tr>
<tr>
<td><code>void SetEntitiesLastFrameInTrigger()</code></td>
<td>Refresh <code>entitiesLastFrameInTrigger</code></td>
</tr>
<tr>
<td><code>void ClearEntities()</code></td>
<td>Clear entity arrays</td>
</tr>
<tr>
<td><code>void CreateCollisionShape()</code></td>
<td>Create static collision</td>
</tr>
<tr>
<td><code>void DestroyCollisionShape()</code></td>
<td>Destroy static collision</td>
</tr>
<tr>
<td><code>void SetActiveStatus(bool status)</code></td>
<td>Set trigger status to active or inactive</td>
</tr>
<tr>
<td><code>bool IsTriggerActive()</code></td>
<td>Check if active flag is set</td>
</tr>
<tr>
<td><code>void SetEntity(const Ptr&lt;Entity&gt; &amp;v)</code></td>
<td>Set entity, this is attached to, to <code>v</code>.</td>
</tr>
<tr>
<td><code>void ClearEntity()</code></td>
<td>Remove entity.</td>
</tr>
<tr>
<td><code>void RegisterMessage(const Id &amp;msgId)</code></td>
<td>Register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

void PhysicsFeature::TriggerProperty::OnRenderDebug() [virtual]

Render a debug visualization of the trigger if needed.

Render a debug visualization of the trigger.

Parameters:
    t transform matrix of my parent rigid body

Reimplemented from Game::Property.

void PhysicsFeature::TriggerProperty::RenderDebugVisualization() [virtual]

Render a debug visualization of the trigger.

Render a debug visualization of the trigger.

Parameters:
    t transform matrix of my parent rigid body

point PhysicsFeature::TriggerProperty::GetTriggerPosition() [protected]

get postion of the trigger

get position of trigger entity

const matrix44 PhysicsFeature::TriggerProperty::GetTriggerTransform() [protected]

get Trigger transform

get transform of the trigger entity

void PhysicsFeature::TriggerProperty::FilterEntities() [protected]
filter current entities in Trigger, this will refresh enterdEntities, insideEntities and leftEntities arrays

cleanup array: enterdEntities, insideEntities, leftEntities and fill them with entity from currEntitiesInTrigger and entities that were in trigger

```cpp
void PhysicsFeature::TriggerProperty::SetCurrentEntitiesInTrigger() [protected]
```

will update the currEntitiesInTrigger

cleanup currEntitiesInTrigger and fill with entities currently in trigger area

```cpp
void PhysicsFeature::TriggerProperty::SetEntitiesLastFrameInTrigger() [protected]
```

refresh entitiesLastFrameInTrigger

method must be invoked at at the end of the frame

```cpp
void PhysicsFeature::TriggerProperty::ClearEntities() [protected]
```

clear entity arrays

reset entity caches, use if trigger was inactive and gets activated

```cpp
void PhysicsFeature::TriggerProperty::SetActiveStatus(bool status) [protected]
```

set trigger status to active or inactive

send a message do it self, some other trigger properties may have interest in the fact that the trigger has been turned off

```cpp
bool PhysicsFeature::TriggerProperty::IsTriggerActive() [protected]
```

check if active flag is set

Render a debug visualization of the trigger.
Parameters:

\[ t \text{ transform matrix of my parent rigid body} \]

```cpp
void Game::Property::OnStart() [virtual, inherited]
```

called from within `Entity::OnStart()` after OnLoad when the complete world exist

This method is called from within `Game::Entity::OnStart()`. This is the moment when the world is complete and the entity can establish connections to other entities.

Reimplemented in `GraphicsFeature::CameraProperty`.

```cpp
void Game::Property::OnMoveBefore() [virtual, inherited]
```

called before movement happens

This method is called from `Game::Entity::OnMoveBefore()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before the physics subsystem is triggered.

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`.

```cpp
void Game::Property::OnMoveAfter() [virtual, inherited]
```

called after movement has happened

This method is called from `Game::Entity::OnMoveAfter()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`, and `PhysicsFeature::PhysicsProperty`.

```cpp
void Game::Property::OnRender() [virtual, inherited]
```
called before rendering happens

This method is called from `Game::Entity::OnRender()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before rendering happens.

Reimplemented in `GraphicsFeature::CameraProperty`, `GraphicsFeature::ChaseCameraProperty`, and `GraphicsFeature::MayaCameraProperty`.

```cpp
void Game::Property::OnLoseActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from `Game::Entity::OnLoseActivity()` on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`.

```cpp
void Game::Property::OnGainActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from `Game::Entity::OnRenderDebug()` on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

```cpp
void Messaging::Port::AttachHandler(const Ptr<Handler> &h) [inherited]
```
attach a message handler to the port

Attach a message handler to the port.

```
void Messaging::Port::RemoveHandler (Ptr<Handler> &h) [inherited]
```

remove a message handler from the port

Remove a message handler from the port.

```
void Messaging::Port::Send (Ptr<Message> &msg) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.

```
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```
void Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields
PIDFeedbackLoop Class Reference

#include <pidfeedbackloop.h>
Detailed Description

A PID feedback loop (proportional integral derivative feedback loop)

(C) 2007 RadonLabs GmbH
PriorityArray Class Reference

#include <priorityarray.h>
Detailed Description

A fixed size priority array. Elements are associated with a priority. New Elements are added to the end of the array until the array is full. In a full array, new elements replace the current lowest priority element (if the priority of the new element is greater of course).

NOTE: The current implementation uses linear search and thus is slow for large arrays.

(C) 2003 RadonLabs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Properties::AmbienceBubbleProperty
Properties::AmbienceBubbleProperty
Class Reference

#include <ambiencebubbleproperty.h>
Detailed Description

Implement a postprocessing "ambience bubble" in the world which changes several postprocessing attributes when the player enters it.

(C) 2006 Radon Labs GmbH
Properties:: BehaviourProperty
Properties::BehaviourProperty Class Reference

#include <behaviourproperty.h>
Detailed Description

Adds state-machine-driven "behaviour" to an entity. Behaviours are defined in XML scripts which define a state machine from standard actions and conditions.

(C) 2006 Radon Labs GmbH
Properties::CutsceneCameraProperty
Properties::CutsceneCameraProperty
Class Reference

#include <cutscenecameraproperty.h>
Detailed Description

A specialized camera property for ingame-cutscenes.

(C) 2007 Radon Labs GmbH
Properties::EnvironmentGraphicsProperty
Properties::EnvironmentGraphicsProperty
Class Reference

#include <environmentgraphicsproperty.h>
Detailed Description

This is a specialized graphics property which handles all the static environment graphics. Probably 90% of all objects in a level will just be static environment objects, so it makes sense to create an optimized class just for that.

NOTE: usually you don't need to care about this class. The level loader will automatically put all static environment objects into a single game entity which has a StaticGraphicsProperty attached.

(C) 2005 Radon Labs GmbH
Properties::InputProperty
Properties::InputProperty Class Reference

#include <inputproperty.h>
**Detailed Description**

An input property adds the ability to handle user input to an entity. If an `InputProperty` is attached to an entity it can become the input focus entity. Global input focus is managed by the `Game::FocusManager` singleton.

If you want the concept of an input focus in your application you should derive your own input property classes from the `InputProperty` class, because then the `FocusManager` will be aware of it (otherwise it will just ignore the entity).

(C) 2007 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Properties::LightProperty
Properties::LightProperty Class Reference

#include <lightproperty.h>
Detailed Description

A light property adds a light source object (Graphics::LightEntity) to a game entity.

(C) 2005 Radon Labs GmbH
Properties::PathAnimProperty
Properties::PathAnimProperty Class Reference

#include <pathanimproperty.h>
Detailed Description

Attach this property to an entity to move the entity along an animation path. Take care that the property won't collide with other properties which influence an entity's position.

(C) 2008 Radon Labs GmbH
Ptr< TYPE > Class Template Reference

#include <ptr.h>
Detailed Description

```cpp
template<class TYPE>
class Ptr< TYPE >
```

Nebula3's smart pointer class which manages the life time of RefCounted objects. Can be used like a normal C++ pointer in most cases.

NOTE: the `Ptr` class is not part of the `Core` namespace for convenience reasons.

(C) 2006 RadonLabs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Ptr()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>Ptr(TYPE *p)</code></td>
<td>construct from C++ pointer</td>
</tr>
<tr>
<td><code>Ptr(const Ptr&lt;TYPE&gt; &amp;p)</code></td>
<td>construct from smart pointer</td>
</tr>
<tr>
<td><code>~Ptr()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void operator=(const Ptr&lt;TYPE&gt; &amp;rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td><code>void operator=(TYPE *rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td><code>bool operator==(const Ptr&lt;TYPE&gt; &amp;rhs) const</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const Ptr&lt;TYPE&gt; &amp;rhs) const</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>bool operator==(const TYPE *rhs) const</code></td>
<td>shortcut equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const TYPE *rhs) const</code></td>
<td>shortcut inequality operator</td>
</tr>
<tr>
<td><code>TYPE * operator-&gt;() const</code></td>
<td>safe -&gt; operator</td>
</tr>
<tr>
<td><code>TYPE &amp; operator*() const</code></td>
<td>safe dereference operator</td>
</tr>
<tr>
<td><code>operator TYPE *() const</code></td>
<td>safe pointer cast operator</td>
</tr>
<tr>
<td>template&lt;class DERIVED&gt; const <code>Ptr&lt; DERIVED &gt; &amp;</code> <code>downcast()</code> const</td>
<td>type-safe downcast operator to other smart pointer</td>
</tr>
<tr>
<td>template&lt;class BASE&gt; const <code>Ptr&lt; BASE &gt; &amp;</code> <code>upcast()</code> const</td>
<td>type-safe upcast operator to other smart pointer</td>
</tr>
<tr>
<td>template&lt;class OTHERTYPE&gt; const <code>Ptr&lt; OTHERTYPE &gt; &amp;</code> <code>cast()</code> const</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><code>isvalid()</code> const</td>
</tr>
<tr>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>TYPE *</td>
<td><code>get()</code> const</td>
</tr>
<tr>
<td>TYPE *</td>
<td><code>get_unsafe()</code> const</td>
</tr>
</tbody>
</table>
Member Function Documentation

template<class TYPE>
template<class OTHERTYPE>
const
OTHERTYPE &
_PTR<>

( ) const
TYPE >::cast

unsafe(!) cast to anything, unless classes have no inheritance-
relationship, call upcast/downcast instead, they are type-safe
QuadTree Class Reference

#include <quadtree.h>
Detailed Description

A simple quad tree. **QuadTree** elements are template nodes and are inserted and removed from a quadtree by bounding box.

(C) 2007 RadonLabs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Render::RenderConsoleHandler
Render::RenderConsoleHandler Class Reference

#include <renderconsolehandler.h>
Detailed Description

Outputs the console. Saves all output texts which go to the standard console and renders it on screen. Allows scrolling with up and down arrow keys.

(C) 2008 Radon Labs GmbH
RenderModules::RenderModule
RenderModules::RenderModule Class Reference

#include <rendermodule.h>

Inheritance diagram for RenderModules::RenderModule:

```
Core::RefCounted

RenderModules::RenderModule
```

Detailed Description

A **RenderModule** wraps a specific, optional, rendering functionality into a simple object which only requires a simple setup. For instance, setting up the debug render functionality in an application looks like this:

```cpp
this->debugRenderModule = DebugRenderModule::Create(); this->debugRenderModule->Setup();
```

This will setup the required environment on the main-thread and render-thread side to implement debug rendering.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RenderModule()</code></td>
<td>() virtual constructor</td>
</tr>
<tr>
<td><code>~RenderModule()</code></td>
<td>() virtual destructor</td>
</tr>
<tr>
<td><code>Setup()</code></td>
<td>() virtual void setup the render module</td>
</tr>
<tr>
<td><code>Discard()</code></td>
<td>() virtual void discard the render module</td>
</tr>
<tr>
<td><code>IsValid()</code></td>
<td>() const return true if the render module has been setup</td>
</tr>
<tr>
<td><code>OnFrame()</code></td>
<td>() virtual void per-frame by Graphics::GraphicsServer</td>
</tr>
<tr>
<td><code>GetRefCount()</code></td>
<td>() const get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>() void increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>() void decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf()</code></td>
<td>(const Rtti &amp;rtti) const return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf()</code></td>
<td>(const Util::String &amp;className) const return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf()</code></td>
<td>(const Util::FourCC &amp;classFourCC) const return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA()</code></td>
<td>(const Rtti &amp;rtti) const return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA()</code></td>
<td>(const Util::String &amp;rttiName) const return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA()</code></td>
<td>(const Util::FourCC &amp;rttiFourCC) const return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td><code>Util::FourCC</code></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class FourCC code</em></td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
 Renders the render module and is a class hierarchy of render elements.

### Data Structures

- Data Structures
- Class Hierarchy
- Data Fields

**RenderModules::RTPlugin**
RenderModules::RTPlugin Class Reference

#include <rtplugin.h>

Inheritance diagram for RenderModules::RTPlugin:
Detailed Description

Standard interface to add new functionality to the render thread. Adding functionality to the render thread usually requires the following steps:

- implement a set of classes which implement the render-thread functionality
- implement proxy classes which act as a frontend in the main thread
- implement a message protocol
- implement a message handler
- derive a new class from RenderThreadPlugin, setup an instance and call GraphicsInterface::Instance()->RegisterRenderThreadPlugin()

Please note that the RenderThreadPlugin object lives completely on the render thread side! Define a clear separation line between main-thread and render-thread code and use messages to communicate between the two!

NOTE: all "On*" methods are called from the RenderThread!

(C) 2009 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTPlugin ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~RTPlugin ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void OnRegister ()</td>
<td>called when plugin is registered on the render-thread side</td>
</tr>
<tr>
<td>virtual void OnUnregister ()</td>
<td>called when plugin is unregistered on the render-thread side</td>
</tr>
<tr>
<td>virtual void OnStageCreated (const Ptr<a href="">InternalGraphics::InternalStage</a> &amp;stage)</td>
<td>called when a new stage has been created</td>
</tr>
<tr>
<td>virtual void OnDiscardStage (const Ptr<a href="">InternalGraphics::InternalStage</a> &amp;stage)</td>
<td>called when a stage is discarded</td>
</tr>
<tr>
<td>virtual void OnViewCreated (const Ptr<a href="">InternalGraphics::InternalView</a> &amp;view)</td>
<td>called when a new view has been created</td>
</tr>
<tr>
<td>virtual void OnDiscardView (const Ptr<a href="">InternalGraphics::InternalView</a> &amp;view)</td>
<td>called when a view is being discarded</td>
</tr>
<tr>
<td>virtual void OnAttachEntity (const Ptr<a href="">InternalGraphics::InternalGraphicsEntity</a> &amp;entity)</td>
<td>called when a graphics entity has been attached to a stage</td>
</tr>
<tr>
<td>virtual void OnRemoveEntity (const Ptr<a href="">InternalGraphics::InternalGraphicsEntity</a> &amp;entity)</td>
<td>called when a graphics entity is being removed from a stage</td>
</tr>
<tr>
<td>virtual void OnUpdateBefore (IndexT framId, Timing::Time time)</td>
<td>called before updating entities</td>
</tr>
<tr>
<td>virtual void OnUpdateAfter (IndexT framId, Timing::Time time)</td>
<td>called after updating entities</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void <code>OnRenderBefore</code> (IndexT frameId, Timing::Time time)</td>
<td>called before rendering entities</td>
</tr>
<tr>
<td>virtual void <code>OnRenderAfter</code> (IndexT frameId, Timing::Time time)</td>
<td>called after rendering entities</td>
</tr>
<tr>
<td>virtual void <code>OnRenderFrameBatch</code> (const Ptr<a href="">Frame::FrameBatch</a> &amp;frameBatch)</td>
<td>called when rendering a frame batch</td>
</tr>
<tr>
<td>virtual void <code>OnRenderWithoutView</code> (IndexT frameId, Timing::Time time)</td>
<td>called if no view exists, and no default camera is set in view</td>
</tr>
<tr>
<td>int <code>GetRefCount</code> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef</code> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release</code> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetClassName</code> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC <code>GetClassFourCC</code> () const</td>
<td></td>
</tr>
</tbody>
</table>
get the class FourCC code
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
RenderModules::RTPluginRegistry
RenderModules::RTPluginRegistry
Class Reference

#include <rtpluginregistry.h>

Inheritance diagram for RenderModules::RTPluginRegistry:
Detailed Description

The central registry for render thread plugins.

(C) 2009 Radon Labs GmbH
Public Member Functions

- **RTPluginRegistry ()**
  - constructor

- **virtual ~RTPluginRegistry ()**
  - destructor

- **void Setup ()**
  - setup the plugin registry

- **void Discard ()**
  - discard the plugin registry

- **bool IsValid () const**
  - return true if currently valid

- **void RegisterRTPlugin (const Core::Rtti *rtti)**
  - register a render plugin

- **void UnregisterRTPlugin (const Core::Rtti *rtti)**
  - unregister a render plugin

- **const Util::Array< Ptr< RTPlugin > > & GetRTPlugins () const**
  - get all currently registered plugins

- **virtual void OnStageCreated (const Ptr< InternalGraphics::InternalStage > &stage)**
  - called when a new stage has been created

- **virtual void OnDiscardStage (const Ptr< InternalGraphics::InternalStage > &stage)**
  - called when a stage is discarded

- **virtual void OnViewCreated (const Ptr< InternalGraphics::InternalView > &view)**
  - called when a new view has been created

- **virtual void OnDiscardView (const Ptr< InternalGraphics::InternalView > &view)**
  - called when a view is being discarded

- **virtual void OnAttachEntity (const Ptr< InternalGraphics::InternalGraphicsEntity... > &entity)**
  - called when an entity is attached
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OnRemoveEntity</code></td>
<td>called when a graphics entity has been registered with the <code>InternalGraphicsServer</code></td>
</tr>
<tr>
<td><code>OnUpdateBefore</code></td>
<td>called before updating entities</td>
</tr>
<tr>
<td><code>OnUpdateAfter</code></td>
<td>called after updating entities</td>
</tr>
<tr>
<td><code>OnRenderBefore</code></td>
<td>called before rendering entities</td>
</tr>
<tr>
<td><code>OnRenderAfter</code></td>
<td>called after rendering entities</td>
</tr>
<tr>
<td><code>OnRenderFrameBatch</code></td>
<td>called from when rendering a frame batch</td>
</tr>
<tr>
<td><code>OnRenderWithoutView</code></td>
<td>called if no view exists, and no default camera is set in view</td>
</tr>
<tr>
<td><code>GetRefCount</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

RenderUtil::DrawFullScreenQuad
RenderUtil::DrawFullScreenQuad Class Reference

#include <drawfullscreenquad.h>
Detailed Description

**Util** class for rendering a full screen quad. Does not care about shader setup!

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DrawFullScreenQuad()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~DrawFullScreenQuad()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>Setup (SizeT rtWidth, SizeT rtHeight)</code></td>
<td>setup the object</td>
</tr>
<tr>
<td><code>Discard()</code></td>
<td>discard the object</td>
</tr>
<tr>
<td><code>IsValid () const</code></td>
<td>return true if object is valid</td>
</tr>
<tr>
<td><code>Draw()</code></td>
<td>draw the fullscreen quad</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:49 2010
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**RenderUtil::MayaCameraUtil**
RenderUtil::MayaCameraUtil Class Reference

#include <mayacamerautil.h>
Detailed Description

Helper class to implement a "Maya camera" with pan/zoom/orbit. Just feed input into the class per its setter methods, call **Update()**, and get the computed view matrix.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MayaCameraUtil()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>void Setup(const Math::point &amp;defaultCenterOfInterest, const Math::point &amp;defaultEyePos, const Math::vector &amp;defaultUpVec)</code></td>
<td>Setup the object</td>
</tr>
<tr>
<td><code>void Reset()</code></td>
<td>Reset the object to its default settings</td>
</tr>
<tr>
<td><code>void Update()</code></td>
<td>Update the view matrix</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetCameraTransform()</code></td>
<td>Get the current camera transform</td>
</tr>
<tr>
<td><code>void SetOrbitButton(bool b)</code></td>
<td>Set state of orbit button</td>
</tr>
<tr>
<td><code>void SetPanButton(bool b)</code></td>
<td>Set state of panning button</td>
</tr>
<tr>
<td><code>void SetZoomButton(bool b)</code></td>
<td>Set state of zoom button</td>
</tr>
<tr>
<td><code>void SetZoomInButton(bool b)</code></td>
<td>Set state of zoom-in button</td>
</tr>
<tr>
<td><code>void SetZoomOutButton(bool b)</code></td>
<td>Set state of zoom-out button</td>
</tr>
<tr>
<td><code>void SetMouseMovement(const Math::float2 &amp;v)</code></td>
<td>Set mouse movement</td>
</tr>
<tr>
<td><code>void SetZoomIn(float v)</code></td>
<td>Set zoom-in value</td>
</tr>
<tr>
<td><code>void SetZoomOut(float v)</code></td>
<td>Set zoom-out value</td>
</tr>
<tr>
<td><code>void SetPanning(const Math::float2 &amp;v)</code></td>
<td>Set panning vector</td>
</tr>
<tr>
<td><code>void SetOrbiting(const Math::float2 &amp;v)</code></td>
<td>Set orbiting vector</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**RenderUtil::MouseRayUtil**
Detailed Description

Helper class to compute a world-space ray from mouse coords.

(C) 2009 Radon Labs GmbH
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static Math::line ComputeWorldMouseRay (const Math::float2 &amp;mousePos, float length, const Math::matrix44 &amp;invViewMatrix, const Math::matrix44 &amp;invProjMatrix, float nearPlane)</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute world-space ray from mouse position (mouse position is expected in the range 0..1)</td>
</tr>
</tbody>
</table>
renderutil::mouseRayUtil::computeWorldMouseRay(const Math::float2 &mousePos, 
float length, 
const Math::matrix44 &invViewMatrix, 
&
const Math::matrix44 &invProjMatrix, 
&
float nearPlane
) [static]

compute world-space ray from mouse position (mouse position is expected in the range 0..1)

Utility function which computes a ray in world space between the eye and the current mouse position on the near plane. Mouse position is expected in the range 0..1.
RenderUtil::NodeLookupUtil
RenderUtil::NodeLookupUtil Class Reference

#include <nodelookuputil.h>
Detailed Description

Helper class to find specific nodes and nodeinstances inside an internalgraphicsentity

WARNING: this util uses SLOW methods, like 'ModelInstanceLookupNodeInstance', use it careful!!!

(C) 2009 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>**static <code>Ptr&lt; Models::StateNodeInstance &gt;** lookupStateNodeInstance (const </code>Ptr<a href="">InternalGraphics::InternalModelEntity</a><code>&amp;entity, const</code>Util::StringAtom` &amp;nodeName, bool checkResourceState=true)</td>
<td>find state node instance</td>
</tr>
<tr>
<td>**static <code>Ptr&lt; Models::AnimatorNodeInstance &gt;** lookupAnimatorNodeInstance (const </code>Ptr<a href="">InternalGraphics::InternalModelEntity</a><code>&amp;entity, const</code>Util::StringAtom` &amp;nodeName, bool checkResourceState=true)</td>
<td>find state node instance</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
Ptr< StateNodeInstance >
RenderUtil::NodeLookupUtil::LookupStateNodeInstance(
    const Ptr< InternalGraphics::InternalModelEntity > &
    const Util::StringAtom &
    bool
)

find state node instance

FIXME: THESE METHODS ARE SLOW AS HELL!!!

Utility function which searches a specified statenodeinstance in a internalgraphicsentity.

returns 0 if

- resource is not loaded and resourcecheck flag is enabled
- node was not found
- internalmodelentity is not active
- model instances is not valid

```cpp
Ptr< AnimatorNodeInstance >
RenderUtil::NodeLookupUtil::LookupAnimatorNodeInstance(
    const Ptr< InternalGraphics::InternalModelEntity > &
    const Util::StringAtom &
    bool
)

find state node instance

FIXME: THESE METHODS ARE SLOW AS HELL!!!

Utility function which searches a specified animatornodeinstance in a internalgraphicsentity.

returns 0 if
```
- resource is not loaded and resourcecheck flag is enabled
- node was not found
- internalmodelentity is not active
- model instanes is not valid
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::D3D9TextureStreamer
Resources::D3D9TextureStreamer
Class Reference

#include <d3d9texturestreamer.h>

Inheritance diagram for Resources::D3D9TextureStreamer:

- Core::RefCounted
- Resources::ResourceLoader
- Resources::StreamResourceLoader
- Resources::D3D9TextureStreamer
- Resources::TextureStreamer
Detailed Description


(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool &amp;SetupResourceFromStream (const Ptr&lt; IO::Stream &gt; &amp;stream)</td>
<td>setup the texture from a Nebula3 stream</td>
</tr>
<tr>
<td>void SetReuseTexture (const Ptr&lt; CoreGraphics::Texture &gt; &amp;tex)</td>
<td>sets the texture to reuse on load</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::Texture &gt; &amp;</td>
<td>GetReuseTexture () const</td>
</tr>
<tr>
<td>get the texture to reuse on load (may return 0!)</td>
<td></td>
</tr>
<tr>
<td>virtual void Reset ()</td>
<td>resets loader-stats e.g. state and reuseTexture (does not cut connection to Resource!)</td>
</tr>
<tr>
<td>virtual bool OnLoadRequested ()</td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td>virtual bool CanLoadAsync () const</td>
<td>override in subclass: return true if asynchronous loading is supported (default is true)</td>
</tr>
<tr>
<td>virtual void OnLoadCancelled ()</td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td>virtual bool OnPending ()</td>
<td>call frequently while after OnLoadRequested() to put Resource into loaded state</td>
</tr>
<tr>
<td>virtual void OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>virtual void OnRemoveFromResource ()</td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>bool IsAttachedToResource () const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetResource () const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td>Resource::State GetState () const</td>
<td>return current state</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td></td>
</tr>
</tbody>
</table>
get the current refcount

void AddRef ()
    increment refcount by one

void Release ()
    decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
    return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
    return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
    return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
    return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
    return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
    return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
    get the class name

Util::FourCC GetClassFourCC () const
    get the class FourCC code
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBLA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual</td>
<td><code>SetupTexture2DFromStream</code> (const <code>Ptr&lt; IO::Stream &gt; &amp;stream)</code></td>
<td>setup a 2D texture from a Nebula3 stream</td>
</tr>
<tr>
<td>virtual</td>
<td><code>SetupTextureCubeFromStream</code> (const <code>Ptr&lt; IO::Stream &gt; &amp;stream)</code></td>
<td>setup a cube texture from a Nebula3 stream</td>
</tr>
<tr>
<td>virtual</td>
<td><code>ReuseMips</code> ()</td>
<td>copies mips from reuseTexture to targeted texture</td>
</tr>
<tr>
<td>virtual</td>
<td><code>LockSurfaces</code> (Util::Array&lt;D3DLOCKED_RECT&gt; &amp;lockedRects, const <code>Ptr&lt; CoreGraphics::Texture &gt; &amp;tex, SizeT numMipsToLock=-1)</code></td>
<td>locks all surfaces of given texture</td>
</tr>
<tr>
<td>virtual</td>
<td><code>UnlockSurfaces</code> (const <code>Ptr&lt; CoreGraphics::Texture &gt; &amp;tex)</code></td>
<td>unlocks all surfaces of given texture</td>
</tr>
<tr>
<td>void</td>
<td><code>GetSurfaceInfo</code> (uint width, uint height, D3DFORMAT fmt, uint *pNumBytes, uint *pRowBytes, uint *pNumRows) const</td>
<td>calculates surface information depending on given width, height and pixel format</td>
</tr>
<tr>
<td>void</td>
<td><code>BitsPerPixel</code> (D3DFORMAT fmt) const</td>
<td>returns appropriate bit-per-pixel-count of given pixel format by looking it up in a table</td>
</tr>
<tr>
<td>uint</td>
<td><code>GetNumRows</code> (uint height, D3DFORMAT fmt) const</td>
<td>returns the number of rows in a line (may depend on format)</td>
</tr>
<tr>
<td>void</td>
<td><code>setState</code> (Resource::State s)</td>
<td>set current state</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
bool Resources::D3D9TextureStreamer::SetupResourceFromStream(const Ptr<IO::Stream> & stream) [virtual]

setup the texture from a Nebula3 stream

This method actually setups the Texture object from the data in the stream.

Reimplemented from `Resources::StreamResourceLoader`.
```

```cpp
bool Resources::D3D9TextureStreamer::OnLoadRequested() [virtual]

called by resource when a load is requested

If we can copy the Texture' data totally from another Texture in memory we don't need to create a file stream.

Reimplemented from `Resources::StreamResourceLoader`.
```

```cpp
bool Resources::D3D9TextureStreamer::SetupTexture2DFromStream(const Ptr<IO::Stream> & stream) [protected, virtual]

setup a 2D texture from a Nebula3 stream

Call this method if something has to be loaded from disk. This may include total reuse of another texture but not partial reuse as this can be done by calling `D3D9TextureStreamer::ReuseMips` which doesn't require a file stream.
```

```cpp
bool Resources::D3D9TextureStreamer::ReuseMips() [protected, virtual]

copies mips from reuseTexture to targeted texture

NOTE: Ensure surfaces which are overwritten by this method are NOT
LOCKED otherwise D3DXLoadSurfaceFromSurface may not work correctly (HRESULT is S_OK but texture is incorrect anyway).

```cpp
void Resources::D3D9TextureStreamer::GetSurfaceInfo ( uint width, uint height, D3DFORMAT fmt, uint * pNumBytes, uint * pRowBytes, uint * pNumRows ) const [protected]
```

calculates surface information depending on given width, height and pixel format.

This method is taken from DirectX SDK Aug 2007 (contained in newer SDKs, too).

```cpp
bool Resources::StreamResourceLoader::CanLoadAsync ( ) const [virtual, inherited]
```

override in subclass: return true if asynchronous loading is supported (default is true).

Indicate whether this resource loader supports asynchronous loading. The default is true. Override this method in a subclass and return false otherwise.

Reimplemented from Resources::ResourceLoader.

Reimplemented in Direct3D9::D3D9StreamShaderLoader.

```cpp
void Resources::StreamResourceLoader::OnLoadCancelled ( ) [virtual, inherited]
```

called by resource to cancel a pending load.

This method is called when the currently pending asynchronous load request should be cancelled.

Reimplemented from Resources::ResourceLoader.
bool
Resources::StreamResourceLoader::OnPending( ) [virtual, inherited]
call frequently while after OnLoadRequested() to put Resource into loaded state
This method is called periodically by the client when the resource is in pending state. The pending ReadStream message will be checked, and if it has been handled successfully the SetupResourceFromStream() method will be called and the Resource will go into Loaded state. If anything goes wrong, the resource will go into Failed state.
Reimplemented from Resources::ResourceLoader.

int
Core::RefCounted::GetRefCount( ) const [inline, inherited]
get the current refcount
Return the current refcount of the object.

void
Core::RefCounted::AddRef( ) [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void
Core::RefCounted::Release( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]
get the class name
Get the class name of the object.
Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
<table>
<thead>
<tr>
<th>Resources</th>
<th>LoadingResource</th>
</tr>
</thead>
</table>

- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields
Resources::LoadingResource Class Reference

#include <loadingresource.h>

Inheritance diagram for Resources::LoadingResource:

```
Core::RefCounted
  |
  V
Resources::LoadingResource
    |
    V
Resources::PoolLoadingResource
```
**Detailed Description**

A **LoadingResource** contains a **Resource** we want to load and a **ManagedResource** which may have a pointer towards a **Resource** we may reuse for loading. ResourceCache keeps an sorted Array of LoadingResources to determine loading-order of requested resources.

(C) 2010 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LoadingResource ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~LoadingResource ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>int GetPriority () const</td>
<td>returns loading-priority-value</td>
</tr>
<tr>
<td>const Ptr&lt; ManagedResource &gt; &amp; GetManagedResource () const</td>
<td>returns ManagedResource</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetTargetResource () const</td>
<td>returns Resource we want to load to</td>
</tr>
<tr>
<td>void SetManagedResource (const Ptr&lt; ManagedResource &gt; &amp; managedResource)</td>
<td>set ManagedResource</td>
</tr>
<tr>
<td>void SetTargetResource (const Ptr&lt; Resource &gt; &amp; targetResource)</td>
<td>set target resource (resource we want to load)</td>
</tr>
<tr>
<td>void SetPriority (int priority)</td>
<td>set the loading-priority</td>
</tr>
<tr>
<td>virtual void OnCancelRequest ()</td>
<td>called as the request is aborted (canceled, aborted, failed, ...)</td>
</tr>
<tr>
<td>virtual void OnSuccessRequest ()</td>
<td>called as the request was performed successfully</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Ptr&lt; ManagedResource &gt;</code></td>
<td><code>managedResource</code></td>
<td>the managed resource we place on the resource after successful loading</td>
</tr>
<tr>
<td><code>Ptr&lt; Resource &gt;</code></td>
<td><code>targetResource</code></td>
<td>the resource we want to load</td>
</tr>
<tr>
<td>bool</td>
<td>operator&lt; (const LoadingResource &amp;a, const LoadingResource &amp;b)</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>less-than operator</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Resources::LoadingResource::OnCancelRequest() [virtual]
called as the request is aborted (canceled, aborted, failed, ...)

Do everything needed in here which has to be performed as a loading request is going to be terminated because it was canceled, aborted, failed and so on.
```

```cpp
void Resources::LoadingResource::OnSuccessRequest() [virtual]
called as the request was performed successfully

Do everything needed in here which has to be performed as a loading request is going to be terminated because it was successful and is no longer needed.
```

Reimplemented in `Resources::PoolLoadingResource`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
```
Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Resources::ManagedMesh
Resources::ManagedMesh Class Reference

#include <managedmesh.h>

Inheritance diagram for Resources::ManagedMesh:
Detailed Description

Specialized managed resource for meshes.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>enum</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority levels</td>
<td></td>
</tr>
</tbody>
</table>
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const Ptr&lt; CoreGraphics::Mesh &gt; &amp; GetMesh () const</code></td>
<td>get contained mesh resource</td>
</tr>
<tr>
<td><code>void ClearRenderStats ()</code></td>
<td>clear render statistics</td>
</tr>
<tr>
<td><code>void UpdateRenderStats (float lod)</code></td>
<td>update render statistics (called by client)</td>
</tr>
<tr>
<td><code>void UpdateRenderStats (const Math::float2 &amp;screenSpaceSize)</code></td>
<td>update render statistics (called by client)</td>
</tr>
<tr>
<td><code>void SetResourceId (const ResourceId &amp;id)</code></td>
<td>set resource id</td>
</tr>
<tr>
<td><code>const ResourceId &amp; GetResourceId () const</code></td>
<td>get resource id</td>
</tr>
<tr>
<td><code>void SetResourceType (const Core::Rtti *rtti)</code></td>
<td>set contained resource type</td>
</tr>
<tr>
<td><code>const Core::Rtti * GetResourceType () const</code></td>
<td>get contained resource type</td>
</tr>
<tr>
<td><code>void IncrClientCount ()</code></td>
<td>increment client count</td>
</tr>
<tr>
<td><code>void DecrClientCount ()</code></td>
<td>decrement client count</td>
</tr>
<tr>
<td><code>SizeT GetClientCount () const</code></td>
<td>get current client count</td>
</tr>
<tr>
<td><code>SizeT GetRenderCount () const</code></td>
<td>get render count for this frame (number of calls to UpdateRenderStats)</td>
</tr>
<tr>
<td><code>float GetResourceStreamingLevelOfDetail () const</code></td>
<td>get resourceStreamingLevelOfDetail for this frame</td>
</tr>
<tr>
<td><code>const Math::float2 &amp; GetMaxScreenSpaceSize () const</code></td>
<td>get maximum screen space size this frame</td>
</tr>
<tr>
<td><code>void</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SetPriority (Priority p)</td>
<td>set current priority</td>
</tr>
<tr>
<td>Priority GetPriority () const</td>
<td>get the current priority</td>
</tr>
<tr>
<td>Resource::State GetState () const</td>
<td>get current resource loading state (Initial, Pending, Loaded, Failed, Cancelled)</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetLoadedResource () const</td>
<td>get contained resource or placeholder if resource is invalid or not loaded</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetResource () const</td>
<td>get contained resource (may return 0)</td>
</tr>
<tr>
<td>bool IsPlaceholder () const</td>
<td>return true if the placeholder resource would be returned</td>
</tr>
<tr>
<td>void Clear ()</td>
<td>clear the contained resource</td>
</tr>
<tr>
<td>bool IsAutoManaged () const</td>
<td>returns true if autoManaged</td>
</tr>
<tr>
<td>void SetAutoManaged (const bool autoManaged)</td>
<td>sets autoManaged-flag</td>
</tr>
<tr>
<td>IndexT GetLastFrameId () const</td>
<td>returns frameld this resource was latest referenced to</td>
</tr>
<tr>
<td>void SetFrameId (const SizeT framId)</td>
<td>sets the frameld the resource was latest used at</td>
</tr>
<tr>
<td>void SetResource (const Ptr&lt; Resource &gt; &amp;resource)</td>
<td>set actual resource</td>
</tr>
<tr>
<td>void SetPlaceholder (const Ptr&lt; Resource &gt; &amp;placeholder)</td>
<td>set placeholder resource</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
void Resources::ManagedResource::ClearRenderStats() [inherited]

clear render statistics

This method resets the current render stats and is usually called during the Prepare() method of the ResourceManager (before rendering is started for the current frame).

void Resources::ManagedResource::UpdateRenderStats(float lod) [inherited]

update render statistics (called by client)

This method is called by the resource client during rendering to write back render statistics. If the resource isn't rendered, the method MUST NOT be called. If the resource is rendered, the client must provide a screen space size guesstimate which will be used by the ResourceMapper to bump or drop the lod of the resource.

void Resources::ManagedResource::UpdateRenderStats(const Math::float2& screenSpaceSize) [inherited]

update render statistics (called by client)

This method is called by the resource client during rendering to write back render statistics. If the resource isn't rendered, the method MUST NOT be called. If the resource is rendered, the client must provide a screen space size guesstimate which will be used by the ResourceMapper to bump or drop the lod of the resource.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.
increment refcount by one

Increment the refcount of the object.

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

get the class name

Get the class name of the object.

get the class FourCC code

Get the class FourCC of the object.

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::ManagedResource
Resources::ManagedResource Class Reference

#include <managedresource.h>

Inheritance diagram for Resources::ManagedResource:
Detailed Description

ManagedResources are wrappers around actual resource objects and are created and managed by the ResourceManager singleton. The actual resource object contained in a ManagedResource may change any time because of the resource management performed by the ResourceManager. During rendering, the resource client writes render-statistics back into the ManagedResource which the resource manager uses as hints for resource management (for instance, if an object is appears very small on screen, the ResourceManager can use this information to drop higher resolution mip levels freeing up valuable memory for other textures).

(C) 2007 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Priority</th>
<th>priority levels</th>
</tr>
</thead>
</table>

### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ManagedResource ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~ManagedResource ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>void ClearRenderStats ()</code></td>
<td>Clear render statistics</td>
</tr>
<tr>
<td><code>void UpdateRenderStats (float lod)</code></td>
<td>Update render statistics (called by client)</td>
</tr>
<tr>
<td><code>void UpdateRenderStats (const Math::float2 &amp;screenSpaceSize)</code></td>
<td>Update render statistics (called by client)</td>
</tr>
<tr>
<td><code>void SetResourceld (const Resourceld &amp;id)</code></td>
<td>Set resource id</td>
</tr>
<tr>
<td><code>const Resourceld &amp; GetResourceld () const</code></td>
<td>Get resource id</td>
</tr>
<tr>
<td><code>void SetResourceType (const Core::Rtti *rtti)</code></td>
<td>Set contained resource type</td>
</tr>
<tr>
<td><code>const Core::Rtti * GetResourceType () const</code></td>
<td>Get contained resource type</td>
</tr>
<tr>
<td><code>void IncrClientCount ()</code></td>
<td>Increment client count</td>
</tr>
<tr>
<td><code>void DecrClientCount ()</code></td>
<td>Decrement client count</td>
</tr>
<tr>
<td><code>SizeT GetClientCount () const</code></td>
<td>Get current client count</td>
</tr>
<tr>
<td><code>SizeT GetRenderCount () const</code></td>
<td>Get render count for this frame (number of calls to <code>UpdateRenderStats()</code>)</td>
</tr>
<tr>
<td><code>float GetResourceStreamingLevelOfDetail () const</code></td>
<td>Get resourceStreamingLevelOfDetail for this frame</td>
</tr>
<tr>
<td><code>const Math::float2 &amp; GetMaxScreenSpaceSize () const</code></td>
<td>Get maximum screen space size this frame</td>
</tr>
<tr>
<td><code>void SetPriority (Priority p)</code></td>
<td>Set priority</td>
</tr>
</tbody>
</table>
set current priority

**Priority**  
*GetPriority() const*
get the current priority

**Resource::State**  
*GetState() const*
get current resource loading state (Initial, Pending, Loaded, Failed, Cancelled)

const **Ptr< Resource > &**  
*GetLoadedResource() const*
get contained resource or placeholder if resource is invalid or not loaded

const **Ptr< Resource > &**  
*GetResource() const*
get contained resource (may return 0)

**bool**  
*IsPlaceholder() const*
return true if the placeholder resource would be returned

**void**  
*Clear()*
clear the contained resource

**bool**  
*IsAutoManaged() const*
returns true if autoManaged

**void**  
*SetAutoManaged(const bool autoManaged)*
sets autoManaged-flag

**IndexT**  
*GetLastFrameId() const*
returns frameId this resource was latest referenced to

**void**  
*SetFrameId(const SizeT frameId)*
sets the frameId the resource was latest used at

**void**  
*SetResource(const **Ptr< Resource > &** resource)*
set actual resource

**void**  
*SetPlaceholder(const **Ptr< Resource > &** placeholder)*
set placeholder resource

**int**  
*GetRefCount() const*
get the current refcount

**void**  
*AddRef()*
increment refcount by one

**void**  
*Release()*
decrement refcount and destroy object if refcount is zero

**bool**  
*IsInstanceOf(const **Rtti** &rtti) const*
return true if this object is instance of given class
<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::String &amp;className) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Resources::ManagedResource::ClearRenderStats()

clear render statistics

This method resets the current render stats and is usually called during the Prepare() method of the ResourceManager (before rendering is started for the current frame).

void
Resources::ManagedResource::UpdateRenderStats(float lod)

update render statistics (called by client)

This method is called by the resource client during rendering to write back render statistics. If the resource isn't rendered, the method MUST NOT be called. If the resource is rendered, the client must provide a screen space size guesstimate which will be used by the ResourceMapper to bump or drop the lod of the resource.

void
Resources::ManagedResource::UpdateRenderStats(const Math::float2& screenSpaceSize)

update render statistics (called by client)

This method is called by the resource client during rendering to write back render statistics. If the resource isn't rendered, the method MUST NOT be called. If the resource is rendered, the client must provide a screen space size guesstimate which will be used by the ResourceMapper to bump or drop the lod of the resource.

int
Core::RefCounted::GetRefCount()

get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::ManagedTexture
Resources::ManagedTexture Class Reference

#include <managedtexture.h>

Inheritance diagram for Resources::ManagedTexture:

```
Core::RefCounted
 |
V
Resources::ManagedResource
 |
V
Resources::ManagedTexture
```
Detailed Description

A specialized managed resource for texture resources.

(C) 2007 Radon Labs GmbH
Public Types

```plaintext
enum Priority
    priority levels
```
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const Ptr&lt; CoreGraphics::Texture &gt; &amp; GetTexture () const</code></td>
<td>get contained texture resource or placeholder if resource is invalid or not loaded</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::Texture &gt; &amp; GetTextureUnloaded () const</code></td>
<td>get contained texture resource and under no circumstances the placeholder - asserts resource is valid!</td>
</tr>
<tr>
<td><code>void ClearRenderStats ()</code></td>
<td>clear render statistics</td>
</tr>
<tr>
<td><code>void UpdateRenderStats (float lod)</code></td>
<td>update render statistics (called by client)</td>
</tr>
<tr>
<td><code>void UpdateRenderStats (const Math::float2 &amp;screenSpaceSize)</code></td>
<td>update render statistics (called by client)</td>
</tr>
<tr>
<td><code>void SetResourceId (const ResourceId &amp;id)</code></td>
<td>set resource id</td>
</tr>
<tr>
<td><code>const ResourceId &amp; GetResourceId () const</code></td>
<td>get resource id</td>
</tr>
<tr>
<td><code>void SetResourceType (const Core::Rtti *rtti)</code></td>
<td>set contained resource type</td>
</tr>
<tr>
<td><code>const Core::Rtti * GetResourceType () const</code></td>
<td>get contained resource type</td>
</tr>
<tr>
<td><code>void IncrClientCount ()</code></td>
<td>increment client count</td>
</tr>
<tr>
<td><code>void DecrClientCount ()</code></td>
<td>decrement client count</td>
</tr>
<tr>
<td><code>SizeT GetClientCount () const</code></td>
<td>get current client count</td>
</tr>
<tr>
<td><code>SizeT GetRenderCount () const</code></td>
<td>get render count for this frame (number of calls to UpdateRenderStats())</td>
</tr>
<tr>
<td><code>float GetResourceStreamingLevelOfDetail</code></td>
<td></td>
</tr>
<tr>
<td>Function/Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>const GetResourceStreamingLevelOfDetail() const</code></td>
<td>Get resource streaming level of detail for this frame</td>
</tr>
<tr>
<td><code>const Math::float2 &amp; GetMaxScreenSpaceSize()</code></td>
<td>Get maximum screen space size this frame</td>
</tr>
<tr>
<td><code>void SetPriority(Priority p)</code></td>
<td>Set current priority</td>
</tr>
<tr>
<td><code>const Priority GetPriority() const</code></td>
<td>Get the current priority</td>
</tr>
<tr>
<td><code>const Resource::State GetState() const</code></td>
<td>Get current resource loading state (Initial, Pending, Loaded, Failed, Cancelled)</td>
</tr>
<tr>
<td><code>const Ptr&lt;Resource&gt; &amp; GetLoadedResource()</code></td>
<td>Get contained resource or placeholder if resource is invalid or not loaded</td>
</tr>
<tr>
<td><code>const Ptr&lt;Resource&gt; &amp; GetResource()</code></td>
<td>Get contained resource (may return 0)</td>
</tr>
<tr>
<td><code>bool IsPlaceholder() const</code></td>
<td>Return true if the placeholder resource would be returned</td>
</tr>
<tr>
<td><code>void Clear()</code></td>
<td>Clear the contained resource</td>
</tr>
<tr>
<td><code>bool IsAutoManaged() const</code></td>
<td>Returns true if autoManaged</td>
</tr>
<tr>
<td><code>void SetAutoManaged(const bool autoManaged)</code></td>
<td>Sets autoManaged-flag</td>
</tr>
<tr>
<td><code>IndexT GetLastFrameId() const</code></td>
<td>Returns frameId this resource was latest referenced to</td>
</tr>
<tr>
<td><code>void SetFrameId(const SizeT frameId)</code></td>
<td>Sets the frameId the resource was latest used at</td>
</tr>
<tr>
<td><code>void SetResource(const Ptr&lt;Resource&gt; &amp;resource)</code></td>
<td>Set actual resource</td>
</tr>
<tr>
<td><code>void SetPlaceholder(const Ptr&lt;Resource&gt; &amp;placeholder)</code></td>
<td>Set placeholder resource</td>
</tr>
</tbody>
</table>
| `int GetRefCount() const` | }
<table>
<thead>
<tr>
<th>Void Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA (const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Getter Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetClassName()</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC()</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Resources::ManagedResource::ClearRenderStats () [inherited]
clear render statistics

This method resets the current render stats and is usually called during the Prepare() method of the ResourceManager (before rendering is started for the current frame).

void Resources::ManagedResource::UpdateRenderStats (float lod ) [inherited]
update render statistics (called by client)

This method is called by the resource client during rendering to write back render statistics. If the resource isn't rendered, the method MUST NOT be called. If the resource is rendered, the client must provide a screen space size guesstimate which will be used by the ResourceMapper to bump or drop the lod of the resource.

void Resources::ManagedResource::UpdateRenderStats (const Math::float2& screenSpaceSize ) [inherited]
update render statistics (called by client)

This method is called by the resource client during rendering to write back render statistics. If the resource isn't rendered, the method MUST NOT be called. If the resource is rendered, the client must provide a screen space size guesstimate which will be used by the ResourceMapper to bump or drop the lod of the resource.

int Core::RefCounted::GetRefCount () const [inline, inherited]
get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Resources::PoolLoadingResource
Resources::PoolLoadingResource
Class Reference

#include <poolloadingresource.h>

Inheritance diagram for Resources::PoolLoadingResource:
Detailed Description

A specialized LoadingResource for pool using strategies.

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PoolLoadingResource ()</strong></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td><strong>~PoolLoadingResource ()</strong></td>
<td><em>destructor</em></td>
</tr>
<tr>
<td><strong>GetNewSlot () const</strong></td>
<td>get target slot (slot we want to load)</td>
</tr>
<tr>
<td><strong>GetOldSlot () const</strong></td>
<td>get the source slot (slot we want to reuse)</td>
</tr>
<tr>
<td><strong>SetTargetSlot (const Ptr&lt; ResourceSlot &gt; &amp; targetSlot)</strong></td>
<td>set target slot (slot we want to load)</td>
</tr>
<tr>
<td><strong>SetSourceSlot (const Ptr&lt; ResourceSlot &gt; &amp; srcSlot)</strong></td>
<td>set the source slot (slot we want to reuse)</td>
</tr>
<tr>
<td><strong>OnSuccessRequest ()</strong></td>
<td>called as the request was performed successfully</td>
</tr>
<tr>
<td><strong>GetPriority () const</strong></td>
<td>returns loading-priority-value</td>
</tr>
<tr>
<td><strong>GetManagedResource () const</strong></td>
<td>returns ManagedResource</td>
</tr>
<tr>
<td><strong>GetTargetResource () const</strong></td>
<td>returns Resource we want to load to</td>
</tr>
<tr>
<td><strong>SetManagedResource (const Ptr&lt; ManagedResource &gt; &amp; managedResource)</strong></td>
<td>set ManagedResource</td>
</tr>
<tr>
<td><strong>SetTargetResource (const Ptr&lt; Resource &gt; &amp; targetResource)</strong></td>
<td>set target resource (resource we want to load)</td>
</tr>
<tr>
<td><strong>SetPriority (int priority)</strong></td>
<td>set the loading-priority</td>
</tr>
<tr>
<td><strong>OnCancelRequest ()</strong></td>
<td>called as the request is aborted (canceled, aborted, failed, ...)</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetClassName</strong> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong> <strong>GetClassFourCC</strong> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ptr&lt; ResourceSlot &gt;</td>
<td>newSlot</td>
<td>the slot containing the resource to load</td>
</tr>
<tr>
<td>Ptr&lt; ResourceSlot &gt;</td>
<td>oldSlot</td>
<td>the slot our ManagedResource’ Resource is currently contained at (may be 0!)</td>
</tr>
<tr>
<td>Ptr&lt; ManagedResource &gt;</td>
<td>managedResource</td>
<td>the managed resource we place on the resource after successful loading</td>
</tr>
<tr>
<td>Ptr&lt; Resource &gt;</td>
<td>targetResource</td>
<td>the resource we want to load</td>
</tr>
</tbody>
</table>
Friends

```cpp
bool operator< (const LoadingResource &a, const LoadingResource &b)
```

less-than operator
Member Function Documentation

void Resources::LoadingResource::OnCancelRequest() [virtual, inherited]
called as the request is aborted (canceled, aborted, failed, ...)
Do everything needed in here which has to be performed as a loading request is going to be terminated because it was canceled, aborted, failed and so on.

int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
get the class name
Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
Resources::PoolResourceMapper
Resources::PoolResourceMapper
Class Reference

#include <poolresourcemapper.h>
Detailed Description

Base-interface for all ResourceCaches using a pool-construct.

(C) 2010 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PoolResourceMapper (void)</td>
<td><em>constructor</em></td>
</tr>
<tr>
<td>~PoolResourceMapper (void)</td>
<td><em>destructor</em></td>
</tr>
<tr>
<td>SetResourceCreatorClass (const Core::Rtti &amp;creatorType)</td>
<td>sets the default loader class</td>
</tr>
<tr>
<td>SetDefaultPoolScheduler (const Ptr&lt;PoolScheduler&gt; &amp;scheduler)</td>
<td>set default scheduler for pools</td>
</tr>
<tr>
<td>virtual void SetupPools (const Util::Dictionary&lt;Util::StringAtom, PoolSetupInfo&gt; &amp;poolSetupData)</td>
<td>creates and initializes pools using given information</td>
</tr>
<tr>
<td>virtual void OnAttachToResourceManager ()</td>
<td>called from resource manager when mapper is attached</td>
</tr>
<tr>
<td>virtual void OnRemoveFromResourceManager ()</td>
<td>called from resource manager when mapper is removed</td>
</tr>
<tr>
<td>virtual void OnPrepare (bool waiting)</td>
<td>called before gathering render stats - updates frameIndex of all pools</td>
</tr>
<tr>
<td>virtual void Reset ()</td>
<td>resets all resource-connections and restores initial-state (keeps pools initialized)</td>
</tr>
<tr>
<td>virtual uint GetAllocatedMemory ()</td>
<td>returns allocated resource space of all pools in bytes</td>
</tr>
<tr>
<td>virtual uint GetUsedMemory ()</td>
<td>returns used memory of all pools in bytes</td>
</tr>
<tr>
<td>virtual void WritePoolsToXML (const IO::URI &amp;fileName)</td>
<td>writes info of all pools to given XML file</td>
</tr>
<tr>
<td>SizeT GetNumPools () const</td>
<td>returns pool count</td>
</tr>
<tr>
<td>GetPoolForDebug (const Util::StringAtom</td>
<td></td>
</tr>
<tr>
<td>const <code>Ptr&lt; ResourcePool &gt; &amp; &amp;poolId</code></td>
<td>(\text{const})</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><code>ReadTexturePoolFromXML</code>&lt;br&gt;(const <code>IO::URI</code> &amp;fileName,&lt;br&gt;<code>Util::Dictionary&lt; Util::StringAtom, PoolSetupInfo &gt;</code> &amp;setupData)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>read given file name and fill given dictionary with data</em></td>
</tr>
<tr>
<td>static void</td>
<td><code>WriteTexturePoolToXML</code>&lt;br&gt;(const <code>IO::URI</code> &amp;fileName, const&lt;br&gt;<code>Util::Dictionary&lt; Util::StringAtom,&lt;br&gt;Resources::PoolSetupInfo &gt;</code> &amp;setupData)</td>
</tr>
<tr>
<td></td>
<td><em>write given pool data to given file name</em></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetFittingPoolIndex</code></td>
<td>(const ResourceInfo *resInfo) const</td>
</tr>
<tr>
<td>returns index of pool which is of given info-type or -1 if no pool found</td>
<td></td>
</tr>
<tr>
<td><code>NoSlotFound</code></td>
<td>(const <code>Ptr&lt; ManagedResource &gt;</code> &amp;resource, IndexT frameIdx)</td>
</tr>
<tr>
<td>this is called if a resource is requested but no free slot is found for</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void
Resources::PoolResourceMapper::OnRemoveFromResourceManager( ) [virtual]
called from resource manager when mapper is removed
unloads all pools, placeholder-resource, ... (shutting down)
```

```cpp
void
Resources::PoolResourceMapper::WritePoolsToXML( const IO::URI & fileName ) [virtual]
writes info of all pools to given XML file
Writes infos of all pools to given XML-file.
```

```cpp
void
Resources::PoolResourceMapper::NoSlotFound( const Ptr< ManagedResource > resource, const IndexT frameIdx ) [protected]
this is called if a resource is requested but no free slot is found for
If no free slot is found we may want to do something like trying to find
next lower or higher level of detail for the requested resource.
Overload this method in subclasses to do things like this.
```

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:49 2010
Resources::PoolSetupInfo
#include <poolresourcemapper.h>
Detailed Description

helper class for pool generation
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::Resource
Resources::Resource Class Reference

#include <resource.h>

Inheritance diagram for Resources::Resource:
Detailed Description

**Base** class for shareable resources. **Resources** contain some sort of data which can be loaded and saved by specialized ResourceLoaders and ResourceSavers.

Before destroying a **Resource** object, the **Loader** and Saver objects must be manually set to null to resolve a cyclic pointer dependency. The **ResourceManager** will take care of this automatically, but when creating resources directly, this must be taken care of!

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>State</th>
</tr>
</thead>
</table>

resource states *(DO NOT CHANGE ORDER!)*
<table>
<thead>
<tr>
<th>Function/Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resource ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~Resource ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void SetAsyncEnabled (bool b)</strong></td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td><strong>bool IsAsyncEnabled () const</strong></td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><strong>void Lock ()</strong></td>
<td>set locked to true</td>
</tr>
<tr>
<td><strong>void Unlock ()</strong></td>
<td>set locked to false</td>
</tr>
<tr>
<td><strong>bool IsLocked () const</strong></td>
<td>returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td><strong>void SetResourceId (const ResourceId &amp;id)</strong></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><strong>const ResourceId &amp; GetResourceId () const</strong></td>
<td>get the resource identifier</td>
</tr>
<tr>
<td><strong>void SetLoader (const Ptr&lt; ResourceLoader &gt; &amp;loader)</strong></td>
<td>set optional resource loader</td>
</tr>
<tr>
<td><strong>const Ptr&lt; ResourceLoader &gt; &amp; GetLoader () const</strong></td>
<td>get optional resource loader</td>
</tr>
<tr>
<td><strong>void SetSaver (const Ptr&lt; ResourceSaver &gt; &amp;saver)</strong></td>
<td>set optional resource saver</td>
</tr>
<tr>
<td><strong>const Ptr&lt; ResourceSaver &gt; &amp; GetSaver () const</strong></td>
<td>get optional resource saver</td>
</tr>
<tr>
<td><strong>SizeT GetUseCount () const</strong></td>
<td>get current use count</td>
</tr>
<tr>
<td><strong>virtual State Load ()</strong></td>
<td>load the resource</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Unload</strong> ()</td>
<td>unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td><strong>SetState</strong> (State s)</td>
<td>set current state (usually only called during Load()!)</td>
</tr>
<tr>
<td><strong>GetState</strong> () const</td>
<td>get current state</td>
</tr>
<tr>
<td><strong>IsLoaded</strong> () const</td>
<td>return true if current state is Loaded</td>
</tr>
<tr>
<td><strong>IsPending</strong> () const</td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td><strong>LoadFailed</strong> () const</td>
<td>return true if current state is Failed</td>
</tr>
<tr>
<td><strong>Save</strong> ()</td>
<td>save the resource</td>
</tr>
<tr>
<td><strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td></td>
</tr>
<tr>
<td>Return type</td>
<td>Method Name</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td>Util::String &amp;</td>
</tr>
<tr>
<td>const</td>
<td></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
</tbody>
</table>

return true if this object is instance of given class, or a derived class, by fourcc
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>IncrUseCount()</code></td>
<td>increment use count</td>
</tr>
<tr>
<td>void</td>
<td><code>DecrUseCount()</code></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Resource::State**

```
Resources::Resource::Load() [virtual]
```

**load the resource**

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the `IsPending()` method will return true as long as the load is in progress, and `IsLoaded()` will become true when the loading process has finished. If the load has failed, `IsPending()` will switch to false and `IsLoaded()` will not be true.

```
void Resources::Resource::Unload() [virtual]
```

**unload the resource, or cancel the pending load**

This will unload the resource. Only call the method when `IsLoaded()` return true. To cancel a pending asynchronous loading process, call the `CancelPendingLoad()` method.

Reimplemented in `CoreAnimation::AnimResource`, `Base::MeshBase`, `Base::VertexBufferBase`, `Direct3D9::D3D9Shader`, `Direct3D9::D3D9Texture`, `Win360::D3D9IndexBuffer`, `Win360::D3D9VertexBuffer`, and `Models::Model`.

```
bool Resources::Resource::Save() [virtual]
```

**save the resource**

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.
int ( ) const [inline, inherited] Core::RefCounted::GetRefCount

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Resources::ResourceDictionary
Resources::ResourceDictionary Class Reference

#include <resourcedictionary.h>

Inheritance diagram for Resources::ResourceDictionary:
Detailed Description

The resource dictionary singleton keeps information about file resources so that the actual resource doesn't have to be loaded first to check its attributes. **Resource** dictionaries are created by asset tools during asset export.

The resource dictionary file format is as follows:

Header: FourCC 'RDIC' - 4 bytes magic number uint version - 4 byte version uint numEntries - 4 byte number of entries

Version 0001 entry (every entry is 128 bytes): uint dataSize - size of the data block of the resource uchar[124] name - 120 bytes block with the resource id, MUST BE 0-TERMINATED!!

**Todo:**

: **ResourceDictionary** should be the base to switch to numerical resource id's later on. During asset export, an offline-dictionary is kept which associated actual filenames with numerical id's, and the asset export tools write those numerical ids in place where a string resource name would be used. For now it's just a lookup-table for the resource size, which is only really useful for console-platforms (since on those platforms the application has more control over the resource loading process as compared to PC-APIs).

(C) 2008 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ResourceDictionary ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~ResourceDictionary ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void SetURI (const IO::URI &amp;uri)</td>
<td>set the dictionary file URI</td>
</tr>
<tr>
<td>const IO::URI &amp; GetURI () const</td>
<td>get the dictionary file URI</td>
</tr>
<tr>
<td>bool Load ()</td>
<td>load resource dictionary from URI</td>
</tr>
<tr>
<td>void Unload ()</td>
<td>unload the current resource dictionary</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>return true if currently loaded (dictionary contains entries)</td>
</tr>
<tr>
<td>bool HasEntry (const ResourceId &amp;resId) const</td>
<td>test if a resource is in the dictionary</td>
</tr>
<tr>
<td>const Entry &amp; GetEntry (const ResourceId &amp;resId) const</td>
<td>get the resource dictionary entry of a resource</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
</tbody>
</table>
return true if this object is instance of given class, or a derived class

```cpp
bool IsA (const Util::String &rttiName) const
    return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
    return true if this object is instance of given class, or a derived class, by fourcc
```

```cpp
const Util::String & GetClassName () const
    get the class name

Util::FourCC GetClassFourCC () const
    get the class FourCC code
```
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
</table>

*dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)*
Data Structures

class Entry

   a resource dictionary entry More...
Member Function Documentation

`bool Resources::ResourceDictionary::Load()`
load resource dictionary from URI

**NOTE:** we assume that the dictionary file is already in host byte order!

`int Core::RefCounted::GetRefCount()` const [inline, inherited]
get the current refcount
Return the current refcount of the object.

`void Core::RefCounted::AddRef()` [inline, inherited]
increment refcount by one
Increment the refcount of the object.

`void Core::RefCounted::Release()` [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

`const Util::String & Core::RefCounted::GetClassName()` const [inline, inherited]
get the class name
Get the class name of the object.

`Util::FourCC Core::RefCounted::GetClassFourCC()` const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
```
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::ResourceDictionary::Entry
Resources::ResourceDictionary::Entry Class Reference

#include <resourcedictionary.h>
Detailed Description

a resource dictionary entry
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Entry()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>void SetSize(SizeT s)</code></td>
<td>set the resource data size</td>
</tr>
<tr>
<td><code>SizeTGetSize() const</code></td>
<td>get the resource data size</td>
</tr>
</tbody>
</table>
Resources::Resourceld Class Reference

#include <resourceid.h>
Detailed Description

ResourceId's are unique identifier of resources used for sharing and locating the resource data on disc.

(C) 2007 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::ResourceLoader
Resources::ResourceLoader Class Reference

#include <resourceloader.h>

Inheritance diagram for Resources::ResourceLoader:
Detailed Description

A resource loader is responsible to setup a resource object with valid data.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function / Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ResourceLoader ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~ResourceLoader ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>OnAttachToResource (const Ptr&lt; Resource &gt;&amp;res)</strong></td>
<td>Called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>virtual void <strong>OnRemoveFromResource ()</strong></td>
<td>Called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>bool <strong>IsAttachedToResource ()</strong> const</td>
<td>Return true if attached to resource</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; Resource &gt; &amp;</strong> <strong>GetResource ()</strong> const</td>
<td>Get pointer to resource</td>
</tr>
<tr>
<td>virtual bool <strong>CanLoadAsync ()</strong> const</td>
<td>Return true if asynchronous loading is supported</td>
</tr>
<tr>
<td>virtual bool <strong>OnLoadRequested ()</strong></td>
<td>Called by resource when a load is requested</td>
</tr>
<tr>
<td>virtual void <strong>OnLoadCancelled ()</strong></td>
<td>Called by resource to cancel a pending load</td>
</tr>
<tr>
<td>virtual bool <strong>OnPending ()</strong></td>
<td>Call frequently while after <strong>OnLoadRequested()</strong> to put <strong>Resource</strong> into loaded state</td>
</tr>
<tr>
<td><strong>Resource::State</strong> <strong>GetState ()</strong> const</td>
<td>Return current state</td>
</tr>
<tr>
<td>virtual void <strong>Reset ()</strong></td>
<td>Resets loader-stats e.g. state</td>
</tr>
<tr>
<td>int <strong>GetRefCount ()</strong> const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName ()</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC ()</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Protected Member Functions

void **SetState** (Resource::State S)

*set current state*
bool Resources::ResourceLoader::CanLoadAsync() const [virtual]

return true if asynchronous loading is supported

This method should be overridden in a subclass and indicates whether the resource loader supports asynchronous resource loading. If asynchronous loading is requested, the OnLoadRequested() method will return immediately and the Resource object will be put into Pending state. Afterwards, the Resource object needs to poll the ResourceLoader using the OnPending method, which will eventually setup the Resource object.

Reimplemented in Direct3D9::D3D9StreamShaderLoader, Models::StreamModelLoader, and Resources::StreamResourceLoader.

bool Resources::ResourceLoader::OnLoadRequested() [virtual]

called by resource when a load is requested

This method is called by our Resource object to perform a synchronous or initiate an asynchronous load. When performing a synchronous load, the method should setup the Resource and go into the Done state (or Failed state when the load has failed). In asynchronous mode, the method should put the resource loader into the Pending state.

Reimplemented in CoreGraphics::MemoryMeshLoader, Win360::D3D9MemoryIndexBufferLoader, Win360::D3D9MemoryVertexBufferLoader, Models::StreamModelLoader, Resources::D3D9TextureStreamer, and Resources::StreamResourceLoader.

void Resources::ResourceLoader::OnLoadCancelled() [virtual]
called by resource to cancel a pending load

This method is called by our `Resource` object if a pending asynchronous load should be cancelled.

Reimplemented in `Models::StreamModelLoader`, and `Resources::StreamResourceLoader`.

```cpp
bool Resources::ResourceLoader::OnPending() [virtual]
```

call frequently while after `OnLoadRequested()` to put `Resource` into loaded state

This method should be called at some time after `OnLoadRequested()` as long as the `ResourceLoader` is in the Pending state. This will check whether the asynchronous loader job has finished, and if yes, setup the `Resource` object, bringing it from the Pending into the Loaded state. If something goes wrong, the `ResourceLoader` will go into the Failed state. If the outstanding loader job isn't finished yet, the `ResourceLoader` should remain in Pending state, and the method should return false. Otherwise the `Resource` should be initialized, and the method should return true.

Reimplemented in `Models::StreamModelLoader`, and `Resources::StreamResourceLoader`.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.
void Core::RefCounted::Release()
[inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName() const
[inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC() const
[inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks()
[static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Resources::ResourceManager
Resources::ResourceManager Class Reference

#include <resourcemanager.h>

Inheritance diagram for Resources::ResourceManager:
Detailed Description

The **ResourceManager** adds a management layer between resource using clients and actual resource objects. The main purpose of the manager is resource streaming for large seamless worlds. **Resource** users request a **ManagedResource** object from the **ResourceManager**. A **ManagedResource** is a wrapper around an actual **Resource** object, which may change based on the resource management strategy implemented by the manager. The main job of a resource manager is to provide all the resource required for rendering while making the best use of limited resource memory. It will also care about background loading of resources, and provide placeholder resources if a resource hasn't finished loading.

While ManagedResources are managed by their appropriate Mappers unmanaged resources are externally created and a reference is stored by the **ResourceManager**. Primarily these are RenderTargets or other **Resources** which are 'never' thrown away.

The actual resource management strategies for different resource types are customizable by attaching **ResourceMapper** objects to the **ResourceManager**. A **ResourceMapper** analyzes the usage statistics of existing **ManagedResource** objects and implements a specific resource management pattern using the following basic operations:

- **Load**(pri, lod): asynchronously load a resource from external storage into memory given a priority and a level-of-detail.
- **Discard**: completely unload a resource, freeing up limited resource memory.
- **Upgrade**(lod): upgrade a loaded resource to a higher level-of-detail
- **Degrade**(lod): degrade a loaded resource to a lower level-of-detail

If **ResourceMapper** is a subclass of StreamingResourceMapper a certain **ResourceScheduler** can be attached on the fly to change management strategy any time.
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ResourceManager()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~ResourceManager()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>void <code>Open()</code></td>
<td>open the resource manager</td>
</tr>
<tr>
<td>void <code>Close()</code></td>
<td>close the resource manager</td>
</tr>
<tr>
<td>bool <code>IsOpen()</code> const</td>
<td>return true if resource manager is open</td>
</tr>
<tr>
<td>void <code>AttachMapper</code> (const <code>Ptr&lt; ResourceManager &gt;</code> &amp;mapper)</td>
<td>register a resource mapper (resource type is defined by mapper)</td>
</tr>
<tr>
<td>void <code>RemoveMapper</code> (const <code>Core::Rtti</code> &amp;resourceType)</td>
<td>unregister a resource mapper by resource type</td>
</tr>
<tr>
<td>void <code>RemoveAllMappers</code></td>
<td>unregister all mappers</td>
</tr>
<tr>
<td>bool <code>HasMapper</code> (const <code>Core::Rtti</code> &amp;resourceType) const</td>
<td>return true if a mapper has been registered for the given resource type</td>
</tr>
<tr>
<td>const <code>Ptr&lt; ResourceManager &gt;</code> &amp;</td>
<td><code>GetMapperByResourceType</code> (const <code>Core::Rtti</code> &amp;resourceType) const</td>
</tr>
<tr>
<td><code>Ptr&lt; ManagedResource &gt;</code></td>
<td><code>CreateManagedResource</code> (const <code>Core::Rtti</code> &amp;resType, const <code>ResourceId</code> &amp;id, const <code>Ptr&lt; ResourceLoader &gt;</code> &amp;optResourceLoader=0)</td>
</tr>
<tr>
<td>void <code>RequestResourceForLoading</code> (const <code>Ptr&lt; ManagedResource &gt;</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>DiscardManagedResource(const <code>Ptr&lt;ManagedResource&gt;</code>&amp; managedResource)</td>
<td>Unloads a managed resource</td>
</tr>
<tr>
<td>HasManagedResource(const <code>ResourceId</code>&amp; id) const</td>
<td>Returns <code>true</code> if a managed resource exists</td>
</tr>
<tr>
<td>LookupManagedResource(const <code>ResourceId</code>&amp; id) const</td>
<td>Looks up a managed resource (does not change usecount of resource)</td>
</tr>
<tr>
<td>AutoManageManagedResource(const <code>ResourceId</code>&amp; id, bool autoManage)</td>
<td>Sets if a given resource whether should be autoManaged or not</td>
</tr>
<tr>
<td>Prepare(bool waiting)</td>
<td>Prepares stats gathering, call per frame</td>
</tr>
<tr>
<td>Update(IndexT frameIdx)</td>
<td>Performs actual resource management, call per frame</td>
</tr>
<tr>
<td>CheckPendingResources()</td>
<td>Tests if any resources are pending, returns <code>true</code> if not</td>
</tr>
<tr>
<td>WaitForPendingResources(Timing::Time timeOut)</td>
<td>Waits until pending resources are loaded, or time-out is reached (returns <code>false</code> if time-out)</td>
</tr>
<tr>
<td>CreateUnmanagedResource(const <code>ResourceId</code>&amp;resId, const <code>Core::Rtti</code>&amp;resClass, const <code>Ptr&lt;ResourceLoader&gt;</code>&amp;loader=0, const <code>Ptr&lt;ResourceSaver&gt;</code>&amp;saver=0)</td>
<td>Creates an unmanaged resource</td>
</tr>
<tr>
<td>RegisterUnmanagedResource(const <code>Ptr&lt;Resource&gt;</code>&amp;res)</td>
<td>Registers an existing resource object as shared resource</td>
</tr>
<tr>
<td>UnregisterUnmanagedResource</td>
<td>Unregisters an existing resource object as shared resource</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>unregister a shared resource (necessary for managing the use count)</code></td>
<td></td>
</tr>
<tr>
<td><code>void UnregisterUnmanagedResource (const ResourceId &amp;id)</code></td>
<td>unregister a shared resource by resource name</td>
</tr>
<tr>
<td><code>void HoldResources ()</code></td>
<td>increments use count of all resources</td>
</tr>
<tr>
<td><code>void ReleaseResources ()</code></td>
<td>decrements use count of all resources</td>
</tr>
<tr>
<td><code>bool HasResource (const ResourceId &amp;id)</code></td>
<td>return true if a shared resource exists</td>
</tr>
<tr>
<td><code>const Ptr&lt;Resource&gt; &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>LookupResource (const ResourceId &amp;id) const</code></td>
<td>lookup a shared resource</td>
</tr>
<tr>
<td><code>Util::Array&lt; Ptr&lt;Resource&gt; &gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>GetResourcesByType (const Core::Rtti &amp;type) const</code></td>
<td>get shared resources by type (slow)</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const <strong>Util::String</strong> &amp;rttiName) const</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::FourCC</strong> &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const <strong>Util::String</strong> &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
Ptr< ManagedResource >
Resources::ResourceManager::CreateManagedResource( const Core::Rtti & resType,
const Resourceld & resId,
const Ptr< ResourceLoader > & optResourceLoader = 0 )
```

create a ManagedResource object (bumps usecount on existing resource)

Create a shared ResourceManager object. If a managed resource with the same resource name already exists, its client count will be incremented and it will be returned. Otherwise the right ResourceMapper will be asked to create a new managed resource.

```cpp
void
Resources::ResourceManager::DiscardManagedResource( const Ptr< ManagedResource > & managedResource )
```

unregister a ManagedResource object

Discard a shared ManagedResource object. This will decrement the client count. If the client count reaches zero, the ManagedResource object will be released as well.

```cpp
void
Resources::ResourceManager::Prepare( bool waiting )
```

prepare stats gathering, call per frame

This method must be called per-frame before rendering begins. This will call the OnPrepare() method on all attached resource mappers, which will at least reset the render statistics in the managed resource.

```cpp
void
Resources::ResourceManager::Update( IndexT frameIdx )
```
perform actual resource management, call per frame

This method must be called by the application after render statistics have been gathered and before the actual rendering. The **ResourceManager** will call the OnUpdate() method on all attached resource mappers. This is the place where the actual resource management will happen.

```cpp
bool Resources::ResourceManager::CheckPendingResources()
```

test if any resources are pending, returns true if not resources are pending

This method tests if there are pending resource which haven't been loaded yet. Returns true if there are no pending resources (everything has loaded), false if pending resources exist.

```cpp
bool Resources::ResourceManager::WaitForPendingResources(Timing::Time timeOut)
```

wait until pending resources are loaded, or time-out is reached (returns false if time-out)

This method blocks until all pending resources are loaded, or until a time-out occurs. If a time-out occurs the method will return false, otherwise true.

```cpp
Ptr<Resource> Resources::ResourceManager::CreateUnmanagedResource(const ResourceId & resId,
const Core::Rtti & resClass,
const Ptr<ResourceLoader> loader = @
const Ptr<ResourceSaver> saver = @
)
```

Create a unmanaged resource object. If the resource already exists, its use count will be increased and the resource will be returned. If the
resource doesn’t exist yet, a new resource object will be created and registered as shared resource. Please note that you must call UnregisterUnmanagedResource() when the resource is no longer needed in order to manage the use count properly.

```cpp
void
Resources::ResourceManager::RegisterUnmanagedResource(const Ptr<Resource>& res)
```

register an existing resource object as shared resource

Register an existing resource object as unmanaged resource. If the resource already has been registered, an assertion will be thrown. This will increment the use count of the resource by one.

```cpp
void
Resources::ResourceManager::UnregisterUnmanagedResource(const Ptr<Resource>& res)
```

unregister a shared resource (necessary for managing the use count)

Unregister an unmanaged resource. This will decrement the use count of the resource. If the use count has reached zero, the resource will be discarded (unloaded and removed from the unmanaged resource pool).

```cpp
void
Resources::ResourceManager::UnregisterUnmanagedResource(const ResourceId& id)
```

unregister a shared resource by resource name

Unregister an unmanaged resource by resource id.

```cpp
bool
Resources::ResourceManager::HasResource(const ResourceId resId)
```

return true if a shared resource exists

Look up resource in registered unmanaged and managed resources.
const Ptr<Resource> & Resources::ResourceManager::LookupResource ( Resourceld resId ) const

lookup a shared resource

Look up resource in registered unmanaged and managed resources.

Array<Ptr<Resource>> Resources::ResourceManager::GetResourcesByType ( Core::Rtti type ) const

get shared resources by type (slow)

Returns an array of unmanaged resources by type. This is a slow method.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name
Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Resources::ResourceMapper
`Resources::ResourceMapper` Class Reference

`#include <resourcemapper.h>`

Inheritance diagram for `Resources::ResourceMapper`:

```
Core::RefCounted

Resources::ResourceMapper

Resources::SimpleResourceMapper
```
Detailed Description

Subclasses of **ResourceMapper** implement specific resource management strategies for one resource type (texture, mesh, etc...). Applications may implement their own specialized ResourceMappers if the provided standard mappers don't fit their needs. ResourceMappers are attached to the **ResourceManager** (one per resource type) and are called back by the resource server to perform resource creation and management. **Resource** clients never talk directly to ResourceMappers, instead they call the **ResourceManager** which in turn talks to the ResourceMappers.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ResourceMapper ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~ResourceMapper ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <strong>SetPlaceholderResourceId</strong> (const ResourceId &amp;resId)</td>
<td>Set a placeholder resource id</td>
</tr>
<tr>
<td>const ResourceId &amp; <strong>GetPlaceholderResourceId ()</strong> const</td>
<td>Get placeholder resource id</td>
</tr>
<tr>
<td>void <strong>SetAsyncEnabled</strong> (bool b)</td>
<td>Set asynchronous behaviour (default is asynchronous)</td>
</tr>
<tr>
<td>bool <strong>IsAsyncEnabled</strong> () const</td>
<td>Return asynchronous loading state</td>
</tr>
<tr>
<td>virtual const Core::Rtti &amp; <strong>GetResourceType</strong> () const</td>
<td>Get resource type handled by this resource mapper</td>
</tr>
<tr>
<td>virtual void <strong>OnAttachToResourceManager ()</strong></td>
<td>Called from resource manager when mapper is attached</td>
</tr>
<tr>
<td>virtual void <strong>OnRemoveFromResourceManager ()</strong></td>
<td>Called from resource manager when mapper is removed</td>
</tr>
<tr>
<td>bool <strong>IsAttachedToResourceManager ()</strong> const</td>
<td>Return true if currently attached to server</td>
</tr>
<tr>
<td>virtual <strong>Ptr&lt; ManagedResource &gt;</strong></td>
<td>OnCreateManagedResource (const Core::Rtti &amp;resType, const ResourceId &amp;resId, const <strong>Ptr&lt; ResourceLoader &gt;</strong> &amp;optResourceLoader)</td>
</tr>
<tr>
<td>virtual void <strong>OnDiscardManagedResource</strong> (const <strong>Ptr&lt; ManagedResource &gt;</strong> &amp;managedResource)</td>
<td>Called when a managed resource should be discarded</td>
</tr>
<tr>
<td>virtual void <strong>OnPrepare</strong> (bool waiting)</td>
<td></td>
</tr>
</tbody>
</table>
called before gathering render stats

```cpp
virtual void OnUpdate (IndexT frameIndex)

called after gathering render stats to perform resource management
```

```cpp
virtual SizeT GetNumPendingResources () const

return the number of currently pending resources
```

```cpp
int GetRefCount () const

get the current refcount
```

```cpp
void AddRef ()

increment refcount by one
```

```cpp
void Release ()

decrement refcount and destroy object if refcount is zero
```

```cpp
bool IsInstanceOf (const Rtti &rtti) const

return true if this object is instance of given class
```

```cpp
bool IsInstanceOf (const Util::String &className) const

return true if this object is instance of given class by string
```

```cpp
bool IsInstanceOf (const Util::FourCC &classFourCC) const

return true if this object is instance of given class by fourcc
```

```cpp
bool IsA (const Rtti &rtti) const

return true if this object is instance of given class, or a derived class
```

```cpp
bool IsA (const Util::String &rttiName) const

return true if this object is instance of given class, or a derived class, by string
```

```cpp
bool IsA (const Util::FourCC &rttiFourCC) const

return true if this object is instance of given class, or a derived class, by fourcc
```

```cpp
const Util::String & GetClassName () const

get the class name
```

```cpp
Util::FourCC GetClassFourCC () const

get the class FourCC code
```
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

`SizeT Resources::ResourceMapper::GetNumPendingResources ( ) const [virtual]`

return the number of currently pending resources

This method must return the number of currently pending resources (resource which have been requested but are not loaded yet).

Reimplemented in `Resources::SimpleResourceMapper`.

`int Core::RefCounted::GetRefCount ( ) const [inline, inherited]`

get the current refcount

Return the current refcount of the object.

`void Core::RefCounted::AddRef ( ) [inline, inherited]`

increment refcount by one

Increment the refcount of the object.

`void Core::RefCounted::Release ( ) [inline, inherited]`

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

`const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]`

get the class name

Get the class name of the object.
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

going the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump recounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::ResourceSaver
#include <resourcesaver.h>

Inheritance diagram for Resources::ResourceSaver:
Detailed Description

A **ResourceSaver** object can write out resource data to a file or another destination. Saving resources is strictly synchronous (unlike loading, which may happen asynchronously).

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resourcesaver ()</strong></td>
<td>virtual constructor</td>
</tr>
<tr>
<td><strong>~Resourcesaver ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</strong></td>
<td>called when the resource saver is attached to its resource</td>
</tr>
<tr>
<td><strong>OnRemoveFromResource ()</strong></td>
<td>called when the resource saver is removed from its resource</td>
</tr>
<tr>
<td><strong>IsAttachedToResource ()</strong> const</td>
<td>boolean return true if attached to resource</td>
</tr>
<tr>
<td><strong>GetResource ()</strong> const</td>
<td>const get pointer to resource</td>
</tr>
<tr>
<td><strong>OnSave ()</strong></td>
<td>virtual boolean return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>GetRefCount ()</strong> const</td>
<td>int get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td>void increment refcount by one</td>
</tr>
<tr>
<td><strong>Release ()</strong></td>
<td>void decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Rtti &amp;rtti)</strong> const</td>
<td>boolean return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf (const Util::String &amp;className)</strong></td>
<td>boolean return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>IsA (const Rtti &amp;rtti)</strong> const</td>
<td>boolean return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>IsA (const Util::String &amp;rttiName)</strong> const</td>
<td>boolean return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>

`class, by string`
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```void
Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```void
Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::ResourceScheduler
Resources::ResourceScheduler Class Reference

#include <resourcescheduler.h>

Inheritance diagram for Resources::ResourceScheduler:

```
Core::RefCounted

Resources::ResourceScheduler

Resources::TexturePoolMapperScheduler
```
Detailed Description

A ResourceScheduler manages all incoming resource requests for a certain StreamingResourceMapper.

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void <code>SetMapper</code></td>
<td>Sets the <code>StreamingResourceMapper</code> to manage</td>
</tr>
<tr>
<td>(const <code>Ptr&lt; StreamingResourceMapper &gt;</code>&amp; mapper)</td>
<td></td>
</tr>
<tr>
<td>virtual const <code>Ptr&lt; LoadingResource &gt;</code> &amp; <code>AppendLoadingResource</code></td>
<td>Appends resource to loading-queue</td>
</tr>
<tr>
<td>(const <code>Core::Rtti</code> &amp;loadingType, const <code>Ptr&lt; ManagedResource &gt;</code> &amp;managedResource, const <code>Ptr&lt; Resource &gt;</code> &amp;resourceToLoad)</td>
<td></td>
</tr>
<tr>
<td>virtual void <code>CleanUpLoadingQueue</code></td>
<td>Cleans the loading queue</td>
</tr>
<tr>
<td>virtual void <code>DoResourceLOD</code></td>
<td>Analyzes and reacts on changes of the distance of given <code>ManagedResource</code></td>
</tr>
<tr>
<td>(const <code>Ptr&lt; ManagedResource &gt;</code> &amp;managedResource)</td>
<td></td>
</tr>
<tr>
<td>virtual bool <code>OnRequestManagedResource</code></td>
<td>Tries to load a resource and returns true if request was successful</td>
</tr>
<tr>
<td>(const <code>Ptr&lt; ManagedResource &gt;</code> &amp;managedResource, const <code>ResourceRequestInfo</code> *requestInfo)</td>
<td></td>
</tr>
<tr>
<td>virtual void <code>OnRemoveFromMapper</code></td>
<td>Call this as the scheduler is removed from its mapper</td>
</tr>
<tr>
<td>int <code>GetRefCount</code> const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef</code> ()</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release</code> ()</td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp; rtttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rtttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
**Member Function Documentation**

```cpp
void Resources::ResourceScheduler::CleanUpLoadingQueue() [virtual]
```

cleans the loading queue

As a very basic this method cleans up loadingQueue and removes loaded Resources. If loading failed or canceled StreamingResourceMapper::ResourceLoadException is called appropriate Resource. Overload this method in subclasses to perform additional clean-ups of the loadingQueue like kicking out requests which are out of date.

```cpp
void Resources::ResourceScheduler::DoResourceLOD(const Ptr<ManagedResource> &managedResource) [v]
```

analyzes and reacts on changes of the distance of given ManagedResource

In subclasses this Method may auto-manage e.g. changement of mip Maps.

Reimplemented in Resources::TexturePoolMapperScheduler.

```cpp
bool Resources::ResourceScheduler::OnRequestManagedResource(const Ptr<ManagedResource> &managed, const ResourceRequestInfo *requestInfo)
```

tries to load a resource and returns true if request was successful

Overwrite this method in subclasses!

Reimplemented in Resources::TexturePoolMapperScheduler.

```cpp
int const [inline, inherited]
```
Core::RefCounted::GetRefCount( )

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::SimpleResourceMapper
Resources::SimpleResourceMapper
Class Reference

#include <simpleresourcemapper.h>

Inheritance diagram for Resources::SimpleResourceMapper:
Detailed Description

Generic, most simple resource mapper which can be used for all types of resources. Asynchronously loads managed resources on demand.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong><code>SimpleResourceMapper()</code></strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~SimpleResourceMapper()`</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>void SetResourceClass(const Core::Rtti &amp;resClass)</strong></td>
<td>Set resource class to be handled (e.g., <code>Mesh::RTTI</code>)</td>
</tr>
<tr>
<td><strong>void SetResourceLoaderClass(const Core::Rtti &amp;resLoaderClass)</strong></td>
<td>Set resource loader class (e.g., <code>Core::Rtti</code>)</td>
</tr>
<tr>
<td><strong>void SetManagedResourceClass(const Core::Rtti &amp;managedResClass)</strong></td>
<td>Set managed resource class (e.g., <code>ManagedMesh::RTTI</code>)</td>
</tr>
<tr>
<td><strong>virtual const Core::Rtti &amp; GetResourceType()`</strong></td>
<td>Get resource type handled by this resource mapper</td>
</tr>
<tr>
<td><strong>virtual void OnAttachToResourceManager()`</strong></td>
<td>Called when mapper is attached</td>
</tr>
<tr>
<td><strong>virtual void OnRemoveFromResourceManager()`</strong></td>
<td>Called when mapper is removed</td>
</tr>
<tr>
<td><strong>virtual <code>Ptr&lt; ManagedResource &gt;</code> OnCreateManagedResource(const Core::Rtti &amp;resType, const ResourceId &amp;resId, const <code>Ptr&lt; ResourceLoader &gt;</code> &amp;optResourceLoader=0)</strong></td>
<td>Create managed resource</td>
</tr>
<tr>
<td><strong>virtual void OnDiscardManagedResource(const <code>Ptr&lt; ManagedResource &gt;</code> &amp;managedResource)</strong></td>
<td>Discard managed resource</td>
</tr>
<tr>
<td><strong>virtual void OnPrepare(bool waiting)</strong></td>
<td>Prepare for rendering</td>
</tr>
<tr>
<td><strong>virtual void OnUpdate(IndexT frameIndex)</strong></td>
<td>Update frame index</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>virtual SizeT GetNumPendingResources () const</code></td>
<td>return the number of currently pending resources</td>
</tr>
<tr>
<td><code>void SetPlaceholderResourceId (const Resourceld &amp;resId)</code></td>
<td>set a placeholder resource id</td>
</tr>
<tr>
<td><code>const Resourceld &amp; GetPlaceholderResourceId () const</code></td>
<td>get placeholder resource id</td>
</tr>
<tr>
<td><code>void SetAsyncEnabled (bool b)</code></td>
<td>set asynchronous behaviour (default is asynchronous)</td>
</tr>
<tr>
<td><code>bool IsAsyncEnabled () const</code></td>
<td>return asynchronous loading state</td>
</tr>
<tr>
<td><code>bool IsAttachedToResourceManager () const</code></td>
<td>return true if currently attached to server</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC)</code> const</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><code>return true if this object is instance of given class, or a derived class, by fourcc</code></td>
</tr>
<tr>
<td>const String</td>
<td><code>GetClassName ()</code> const</td>
</tr>
<tr>
<td></td>
<td><code>get the class name</code></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><code>GetClassFourCC ()</code> const</td>
</tr>
<tr>
<td></td>
<td><code>get the class FourCC code</code></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
Ptr< ManagedResource >
Resources::SimpleResourceMapper::OnCreateManagedResource ( const Core::Rtti & resType,
const Resourceld & resId,
const Ptr< ResourceLoader > & optResourceLoader = 0
)
```
called when a managed resource should be created

This method is called by the `ResourceManager` when a new ManagedResource must be created. The resource manager will only call this method if this is the first request for the resource name.

Reimplemented from `Resources::ResourceMapper`.

```cpp
void
Resources::SimpleResourceMapper::OnDiscardManagedResource ( const Ptr< ManagedResource > & managedResource
)
```
called when a managed resource should be discarded

This method will be called by the `ResourceManager` whenever a ManagedResource should be discarded.

Reimplemented from `Resources::ResourceMapper`.

```cpp
void
Resources::SimpleResourceMapper::OnPrepare ( bool waiting ) [virtual]
```
called before gathering render stats

This method will go through all ManagedResources and reset their render statistics. It will also check whether pending resources have finished loading, and update the associated managed resources accordingly.

Reimplemented from `Resources::ResourceMapper`.
int const Core::RefCounted::GetRefCount() [inline, inherited]

get the current refcount

Return the current refcount of the object.

void const Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void const Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & const Core::RefCounted::GetClassName() [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC const Core::RefCounted::GetClassFourCC() [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application
exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::StreamResourceLoader
Resources::StreamResourceLoader
Class Reference

#include <streamresourceloader.h>

Inheritance diagram for Resources::StreamResourceLoader:
Detailed Description

**Base** class for resource loaders which load their resource from an **IO::Stream**. Handles the details of synchronous vs. asynchronous resource loading. Subclasses only need to override the **SetupResourceFromStream()** virtual method.

(C) 2008 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StreamResourceLoader ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~StreamResourceLoader ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual bool <code>CanLoadAsync ()</code> const</td>
<td>override in subclass: return true if asynchronous loading is supported (default is true)</td>
</tr>
<tr>
<td>virtual bool <code>OnLoadRequested ()</code></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td>virtual void <code>OnLoadCancelled ()</code></td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td>virtual bool <code>OnPending ()</code></td>
<td>call frequently while after <code>OnLoadRequested()</code> to put <code>Resource</code> into loaded state</td>
</tr>
<tr>
<td>virtual void <code>OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</code></td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>virtual void <code>OnRemoveFromResource ()</code></td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>bool <code>IsAttachedToResource ()</code> const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; <code>GetResource ()</code> const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td><code>Resource::State</code> <code>GetState ()</code> const</td>
<td>return current state</td>
</tr>
<tr>
<td>virtual void <code>Reset ()</code></td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td>int <code>GetRefCount ()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Rtti &amp;rtti)</code> const</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;className) const</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>virtual bool</th>
<th><strong>SetupResourceFromStream</strong> (const Ptr&lt; IO::Stream &gt;&amp;stream)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>override in subclass: setup the resource from the provided stream</em></td>
</tr>
<tr>
<td>void</td>
<td><strong>SetState</strong> <em>(Resource::State s)</em></td>
</tr>
<tr>
<td></td>
<td><em>set current state</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool Resources::StreamResourceLoader::CanLoadAsync() const [virtual]
```

override in subclass: return true if asynchronous loading is supported (default is true)

Indicate whether this resource loader supports asynchronous loading. The default is true. Override this method in a subclass and return false otherwise.

Reimplemented from `Resources::ResourceLoader`.

Reimplemented in `Direct3D9::D3D9StreamShaderLoader`.

```cpp
bool Resources::StreamResourceLoader::OnLoadRequested() [virtual]
```

called by resource when a load is requested

Handle the generic load request. In the asynchronous case, this method will fire a ReadStream message and go into pending state. The client will then call `OnPending()` periodically which checks whether the ReadStream message has been handled and continue accordingly. In the synchronous state the method will create an `IO::Stream` object and call the `SetupResourceFromStream()` method directly.

Reimplemented from `Resources::ResourceLoader`.

Reimplemented in `Resources::D3D9TextureStreamer`.

```cpp
void Resources::StreamResourceLoader::OnLoadCancelled() [virtual]
```

called by resource to cancel a pending load

This method is called when the currently pending asynchronous load
request should be cancelled.

Reimplemented from Resources::ResourceLoader.

```cpp
bool Resources::StreamResourceLoader::OnPending() [virtual]
```

call frequently while after OnLoadRequested() to put Resource into loaded state

This method is called periodically by the client when the resource is in pending state. The pending ReadStream message will be checked, and if it has been handled successfully the SetupResourceFromStream() method will be called and the Resource will go into Loaded state. If anything goes wrong, the resource will go into Failed state.

Reimplemented from Resources::ResourceLoader.

```cpp
bool Resources::StreamResourceLoader::SetupResourceFromStream(const Ptr<IO::Stream> & stream) [protected, virtual]
```

override in subclass: setup the resource from the provided stream

The SetupResourceFromStream should be overwritten by the subclass and is called when the stream containing the resource data is available (right from OnLoadRequested() in the synchronous state, or at some later time from OnPending() in the asynchronous state. Subclasses should setup the resource object of this loader from the stream content and return true on success, or false on failure.

Reimplemented in CoreAnimation::StreamAnimationLoader, and Resources::D3D9TextureStreamer.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::TexturePoolMapperScheduler
Resources::TexturePoolMapperScheduler
Class Reference

#include <texturepoolmapperscheduler.h>

Inheritance diagram for Resources::TexturePoolMapperScheduler:

```
Core::RefCounted

Resources::ResourceScheduler

Resources::TexturePoolMapperScheduler
```
Detailed Description

A simple (example) for a ResourceScheduler for TexturePools based on NRU (not recently used) algorithm.

(C) 2010 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TexturePoolMapperScheduler ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~TexturePoolMapperScheduler ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>SetMapper (const Ptr&lt; StreamingResourceMapper &gt; &amp;mapper)</strong></td>
<td>sets the ResourceMapper and the PoolResourceMapper (downcasts the ResourceMapper)</td>
</tr>
<tr>
<td><strong>DoResourceLOD (const Ptr&lt; ManagedResource &gt; &amp;managedResource)</strong></td>
<td>analyzes and reacts on changes of the distance of given ManagedResource</td>
</tr>
<tr>
<td>*<em>OnRequestManagedResource (const Ptr&lt; ManagedResource &gt; &amp;managedResource, const ResourceRequestInfo <em>requestInfo)</em></em></td>
<td>tries to load a resource and returns true if request was successful</td>
</tr>
<tr>
<td><strong>OnRemoveFromMapper ()</strong></td>
<td>call this as the scheduler is removed from its mapper</td>
</tr>
<tr>
<td><strong>AppendLoadingResource (const Core::Rtti &amp;loadingType, const Ptr&lt; ManagedResource &gt; &amp;managedResource, const Ptr&lt; Resource &gt; &amp;resourceToLoad)</strong></td>
<td>appends resource to loading-queue</td>
</tr>
<tr>
<td><strong>CleanUpLoadingQueue ()</strong></td>
<td>cleans the loading queue</td>
</tr>
<tr>
<td><strong>GetRefCount ()</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef ()</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
**Protected Member Functions**

<table>
<thead>
<tr>
<th>virtual bool</th>
<th><code>OnRequestOtherMipMap</code> (const <code>Ptr&lt; ManagedTexture &gt;</code> &amp;managedTexture, const TextureRequestInfo *requestInfo)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tries to copy as much texture data from texture in memory to another slot fitting new mipMap level</td>
</tr>
</tbody>
</table>
Protected Attributes

Ptr< PoolResourceMapper > poolMapper
void Resources::TexturePoolMapperScheduler::SetMapper ( const Ptr< StreamingResourceMapper > & mapper ) [virtual]

sets the ResourceMapper and the PoolResourceMapper (downcasts the ResourceMapper)

This stores a downcasted copy of the Ptr as a separate PoolResourceMapper so we don't need to cast it frequently OnFrame.

Reimplemented from Resources::ResourceScheduler.

bool Resources::TexturePoolMapperScheduler::OnRequestManagedResource ( const Ptr< ManagedResource > & managedResource, const ResourceRequestInfo * requestInfo )

tries to load a resource and returns true if request was successful

Looks up a fitting pool and a free slot in this pool is looked up. If there is a free slot available the slot's Resource is set to requested Resource and a new LoadingResource-entry is sorted into the loadingQueue. If NO free slot is currently available false is returned. If NO free pool is found an error is thrown as this must not occur as each requested resource should have a valid entry in the resource dictionary.

Reimplemented from Resources::ResourceScheduler.

void Resources::ResourceScheduler::CleanUpLoadingQueue ( ) [virtual, inherited]

cleans the loading queue
As a very basic this method cleans up loadingQueue and removes loaded **Resources**. If loading failed or canceled `StreamingResourceMapper::ResourceLoadException` is called appropriate **Resource**. Overload this method in subclasses to perform additional clean-ups of the loadingQueue like kicking out requests which are out of date.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Field Documentation

`Ptr<PoolResourceMapper> Resources::TexturePoolMapperScheduler::poolMapper` [protected]

points to same target as ResourceScheduler::mapper but we do need this as we want to have some `PoolResourceMapper` specific functionalities
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::TextureStreamer
Resources::TextureStreamer Class Reference

#include <texturestreamer.h>

Inheritance diagram for Resources::TextureStreamer:
Detailed Description


(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual bool SetupResourceFromStream(const Ptr&lt; IO::Stream &gt; &amp;stream)</code></td>
<td>setup the texture from a Nebula3 stream</td>
</tr>
<tr>
<td><code>void SetReuseTexture(const Ptr&lt; CoreGraphics::Texture &gt; &amp;tex)</code></td>
<td>sets the texture to reuse on load</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::Texture &gt; &amp; GetReuseTexture() const</code></td>
<td>get the texture to reuse on load (may return 0!)</td>
</tr>
<tr>
<td><code>virtual void Reset()</code></td>
<td>resets loader-stats e.g. state and reuseTexture (does not cut connection to Resource!)</td>
</tr>
<tr>
<td><code>virtual bool OnLoadRequested()</code></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td><code>virtual bool CanLoadAsync() const</code></td>
<td>override in subclass: return true if asynchronous loading is supported (default is true)</td>
</tr>
<tr>
<td><code>virtual void OnLoadCancelled()</code></td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td><code>virtual void OnPending()</code></td>
<td>call frequently while after <code>OnLoadRequested()</code> to put Resource into loaded state</td>
</tr>
<tr>
<td><code>virtual void OnAttachToResource(const Ptr&lt; Resource &gt; &amp;res)</code></td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td><code>virtual void OnRemoveFromResource()</code></td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td><code>bool IsAttachedToResource() const</code></td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td><code>const Ptr&lt; Resource &gt; &amp; GetResource() const</code></td>
<td>get pointer to resource</td>
</tr>
<tr>
<td><code>Resource::State GetState() const</code></td>
<td>return current state</td>
</tr>
<tr>
<td><code>int GetRefCount() const</code></td>
<td></td>
</tr>
</tbody>
</table>
get the current refcount

void **AddRef ()
// increment refcount by one

void **Release ()
// decrement refcount and destroy object if refcount is zero

bool **IsInstanceOf (const Rtti &rtti) const
// return true if this object is instance of given class

bool **IsInstanceOf (const Util::String &className) const
// return true if this object is instance of given class by string

bool **IsInstanceOf (const Util::FourCC &classFourCC) const
// return true if this object is instance of given class by fourcc

bool **IsA (const Rtti &rtti) const
// return true if this object is instance of given class, or a derived class

bool **IsA (const Util::String &rttiName) const
// return true if this object is instance of given class, or a derived class, by string

bool **IsA (const Util::FourCC &rttiFourCC) const
// return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & **GetClassName () const
// get the class name

Util::FourCC **GetClassFourCC () const
// get the class FourCC code
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetupTexture2DFromStream (const Ptr&lt; IO::Stream &gt;&amp;stream)</code></td>
<td>setup a 2D texture from a Nebula3 stream</td>
</tr>
<tr>
<td><code>SetupTextureCubeFromStream (const Ptr&lt; IO::Stream &gt;&amp;stream)</code></td>
<td>setup a cube texture from a Nebula3 stream</td>
</tr>
<tr>
<td><code>ReuseMips ()</code></td>
<td>copies mips from reuseTexture to targeted texture</td>
</tr>
<tr>
<td><code>LockSurfaces (Util::Array&lt; D3DLOCKED_RECT &gt;&amp;lockedRects, const Ptr&lt; CoreGraphics::Texture &gt;&amp;tex, SizeT numMipsToLock=-1)</code></td>
<td>locks all surfaces of given texture</td>
</tr>
<tr>
<td><code>UnlockSurfaces (const Ptr&lt; CoreGraphics::Texture &gt;&amp;tex)</code></td>
<td>unlocks all surfaces of given texture</td>
</tr>
<tr>
<td><code>GetSurfaceInfo (uint width, uint height, D3DFORMAT fmt, uint *pNumBytes, uint *pRowBytes, uint *pNumRows)</code></td>
<td>calculates surface information depending on given width, height and pixel format</td>
</tr>
<tr>
<td><code>BitsPerPixel (D3DFORMAT fmt) const</code></td>
<td>returns appropriate bit-per-pixel-count of given pixel format by looking it up in a table</td>
</tr>
<tr>
<td><code>GetNumRows (uint height, D3DFORMAT fmt) const</code></td>
<td>returns the number of rows in a line (may depend on format)</td>
</tr>
<tr>
<td><code>SetState (Resource::State s)</code></td>
<td>set current state</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
bool Resources::D3D9TextureStreamer::SetupResourceFromStream(const Ptr<IO::Stream> &stream) [virtual, inherited]
```

setup the texture from a Nebula3 stream

This method actually sets the Texture object from the data in the stream.

Reimplemented from **Resources::StreamResourceLoader**.

```cpp
bool Resources::D3D9TextureStreamer::OnLoadRequested() [virtual, inherited]
```

called by resource when a load is requested

If we can copy the Texture' data totally from another Texture in memory we don't need to create a file stream.

Reimplemented from **Resources::StreamResourceLoader**.

```cpp
bool Resources::D3D9TextureStreamer::SetupTexture2DFromStream(const Ptr<IO::Stream> &stream) [protected, virtual, inherited]
```

setup a 2D texture from a Nebula3 stream

Call this method if something has to be loaded from disk. This may include total reuse of another texture but not partial reuse as this can be done by calling **D3D9TextureStreamer::ReuseMips** which doesn't require a file stream.

```cpp
bool Resources::D3D9TextureStreamer::ReuseMips() [protected, virtual, inherited]
```

copies mips from reuseTexture to targeted texture

**NOTE:** Ensure surfaces which are overwritten by this method are NOT
LOCKED otherwise D3DXLoadSurfaceFromSurface may not work correctly (HRESULT is S_OK but texture is incorrect anyway).

void
Resources::D3D9TextureStreamer::GetSurfaceInfo(uint width,
uint height,
D3DFORMAT fmt,
uint * pNumBytes,
uint * pRowBytes,
uint * pNumRows)
const [protected, inherited]

calculates surface information depending on given width, height and pixel format

This method is taken from DirectX SDK Aug 2007 (contained in newer SDKs, too).

bool
Resources::StreamResourceLoader::CanLoadAsync() const [virtual, inherited]

override in subclass: return true if asynchronous loading is supported (default is true)

Indicate whether this resource loader supports asynchronous loading. The default is true. Override this method in a subclass and return false otherwise.

Reimplemented from Resources::ResourceLoader.

Reimplemented in Direct3D9::D3D9StreamShaderLoader.

void
Resources::StreamResourceLoader::OnLoadCancelled() [virtual, inherited]

called by resource to cancel a pending load

This method is called when the currently pending asynchronous load request should be cancelled.
Reimplemented from Resources::ResourceLoader.

```cpp
bool Resources::StreamResourceLoader::OnPending() [virtual, inherited]
```
call frequently while after OnLoadRequested() to put Resource into loaded state

This method is called periodically by the client when the resource is in pending state. The pending ReadStream message will be checked, and if it has been handled successfully the SetupResourceFromStream() method will be called and the Resource will go into Loaded state. If anything goes wrong, the resource will go into Failed state.

Reimplemented from Resources::ResourceLoader.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name
Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Round Class Reference

#include <round.h>
**Detailed Description**

Helper class for rounding up integer values to $2^N$ values.

(C) 2009 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Script::ActionReader
#include <actionreader.h>

Inheritance diagram for Script::ActionReader:

```
Core::RefCounted

Script::ActionReader
```
Detailed Description

Provides functionality to read and write masterEvents and conditions from and to strings.

Note: Be aware that SetString will already tokenize the given string and therefore a lot of the work is done here (dont call it too often).

(C) 2006 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Public Member Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ActionReader ()</strong></td>
</tr>
<tr>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~<strong>ActionReader ()</strong></td>
</tr>
<tr>
<td>destructor</td>
</tr>
<tr>
<td>int <strong>GetRefCount ()</strong> const</td>
</tr>
<tr>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
</tr>
<tr>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
</tr>
<tr>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetClassName ()</strong> const</td>
</tr>
<tr>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong> <strong>GetClassFourCC ()</strong> const</td>
</tr>
<tr>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>void <strong>SetString</strong> (const Util::String &amp;actionString)</td>
</tr>
<tr>
<td>set the string to be parsed (read mode)</td>
</tr>
</tbody>
</table>
Ptr< Actions::Action > GetAction ()
generate an action from the string (read mode)

Ptr< Conditions::Condition > GetCondition ()
generate an action from the string (read mode)

Util::String GetClass ()
get classname

int GetInt ()
get an int

bool GetBool ()
get a bool

float GetFloat ()
get a float

Util::String GetString ()
get a string

Ptr< Game::Entity > GetEntity ()
get an entity (might be 0)

Math::vector GetVector ()
get a vector3

Math::float4 GetFloat4 ()
get a vector4

int Version () const
return content version of the action string

const Util::String & GetActionString () const
get the action string (write mode)

void PutClass (Ptr< Core::RefCounted > &refcounted)
put the classname

void PutInt (int value)
put an int

void PutBool (bool value)
put a bool

void PutFloat (float value)
put a float

void PutString (const Util::String &text)
put a string
```plaintext
void PutEntity (Ptr< Game::Entity > entity)
    put an entity (might be 0)

void PutVector (const Math::vector &vec)
    put a vector3

void PutFloat4 (const Math::float4 &vec)
    put a vector4

void PutVersion (int version)
    put
```
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Util::String NextToken (const Util::String &amp;sourceString, int &amp;startIndex, char limiter)</code></td>
<td>helper function to extract data from string until limiter</td>
</tr>
<tr>
<td><code>Util::String NextToken (const Util::String &amp;sourceString, int &amp;startIndex)</code></td>
<td>helper function to extract data from string until &quot; &quot; or &quot;;&quot;</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
get the current refcount
Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
get the class name
Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Script::Attributes
Script::Attributes Class Reference

#include <scriptattributes.h>
Detailed Description

Declare attributes used by the story subsystem system.

(C) 2005 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Script::Dialog
Script::Dialog Class Reference

#include <dialog.h>

Inheritance diagram for Script::Dialog:

```
Core::RefCounted
  
Script::Dialog
```
Detailed Description

Contains the runtime structure of a complete dialog.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dialog ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~Dialog ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual <strong>Ptr&lt; DialogTake &gt; CreateDialogTake ()</strong></td>
<td>create a Take (override if needed)</td>
</tr>
<tr>
<td>void <strong>LoadDialogContent (Ptr&lt; DialogDesc &gt; dialogDesc)</strong></td>
<td>load dialog content, takes and so on, after that Dialog content is usable (param group is optional)</td>
</tr>
<tr>
<td>bool <strong>ActivateDialog (const Ptr&lt; DialogTake &gt; &amp;startTake=0)</strong></td>
<td>activates the dialog</td>
</tr>
<tr>
<td>void <strong>DeactivateDialog ()</strong></td>
<td>deactivates the dialog</td>
</tr>
<tr>
<td>const <strong>Util::Guid &amp; GetGuid () const</strong></td>
<td>get dialog guid</td>
</tr>
<tr>
<td>const <strong>Util::String &amp; GetId () const</strong></td>
<td>get dialog id</td>
</tr>
<tr>
<td>const <strong>Util::String &amp; GetSpeaker () const</strong></td>
<td>get dialog speaker</td>
</tr>
<tr>
<td>const <strong>Util::String &amp; GetGroup () const</strong></td>
<td>get dialog speaker group</td>
</tr>
<tr>
<td>bool <strong>IsLoaded () const</strong></td>
<td>return true if a dialog is loaded</td>
</tr>
<tr>
<td>bool <strong>IsActive () const</strong></td>
<td>return true if a dialog was activated</td>
</tr>
<tr>
<td>bool <strong>IsSpeaker (const Util::String &amp;name) const</strong></td>
<td>check if someone is a possible speaker (doesn't need to actually say anything!)</td>
</tr>
<tr>
<td>void <strong>setCurrentTake (const Ptr&lt; DialogTake &gt; &amp;take)</strong></td>
<td>set current Dialog Take Pointer</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const Ptr&lt; DialogTake &gt; &amp; GetCurrentTake() const</code></td>
<td>Get current <code>Dialog</code> Take Pointer, the dialog must be active</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Util::String &gt; &amp; GetDialogTexts() const</code></td>
<td>Get current take text array</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Util::String &gt; &amp; GetResponseTexts() const</code></td>
<td>Get response text array</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Util::String &gt; &amp; GetShortResponseTexts() const</code></td>
<td>Get short response text array</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetTakeSoundID() const</code></td>
<td>Get the CueId of the current take</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Util::String &gt; &amp; GetResponseSoundIDs() const</code></td>
<td>Get a array with the CueIds of the current responses</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetTakeSpeaker() const</code></td>
<td>Get the Speaker of the current take</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Util::String &gt; &amp; GetResponseSpeakers() const</code></td>
<td>Get a array with the Speaker's of the current responses</td>
</tr>
<tr>
<td><code>const Util::Array&lt; bool &gt; &amp; GetResponsePassedStates() const</code></td>
<td>Gets a array with the passed state of the current responses</td>
</tr>
<tr>
<td><code>const Ptr&lt; DialogTake &gt; &amp; GetResponseAtIndex(int index) const</code></td>
<td>Get index of Storage Take in Response</td>
</tr>
<tr>
<td><code>bool HasNextTakeLayer(const Ptr&lt;DialogTake&gt; &amp;responseTake)</code></td>
<td>Has a following layer, an NPC Take, in the dialog hierarchy (FIXME Floh:???)</td>
</tr>
<tr>
<td><code>Ptr&lt; DialogTake &gt; NavigateToNextLayer(const Ptr&lt;DialogTake&gt; &amp;responseTake)</code></td>
<td>Navigates to the next layer in the dialog hierarchy, returns pointer to next <code>DialogTake</code></td>
</tr>
<tr>
<td><code>void UpdateDialogData()</code></td>
<td>Updates the dialog-data, refresh ResponseTexts, DialogTexts and ResponseIndex</td>
</tr>
<tr>
<td><code>bool AssertDialog()</code></td>
<td>Executes the assert function from the take’s of the current dialog</td>
</tr>
<tr>
<td><code>Ptr&lt; DialogTake &gt; GetTakeById(const Util::String &amp;id)</code></td>
<td>Returns a take by id</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void AddRef ()</strong></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>bool IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>bool IsA (const Util::String &amp;rttiName) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>bool IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetClassName () const</strong></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC GetClassFourCC () const</strong></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

```cpp
void Script::Dialog::LoadDialogContent (Ptr<DialogDesc> desc)

load dialog content, takes and so on, after that Dialog content is usable (param group is optional)

Load the dialog's content from the database.

bool Script::Dialog::ActivateDialog (const Ptr<DialogTake> &startTake = 0)

activates the dialog

Activates the current Dialog. That means, we set the pointer to the first valid DialogTake in the start take list, where valid means evaluation of take condition returns true. Activation will fail, and return false if dialog has no valid DialogTakes If a start take is given, set it as start take.

bool Script::Dialog::HasNextTakeLayer (const Ptr<DialogTake> &responseTake)

has a following layer, an NPC Take, in the dialog hierarchy (FIXME Floh:????)

navigates to the next layer in the dialog hierarchy may return 0 if no following take is found

Ptr<DialogTake> Script::Dialog::NavigateToNextLayer (const Ptr<DialogTake> &responseTake)

navigates to the next layer in the dialog hierarchy, returns pointer to next DialogTake

navigates to the next layer in the dialog hierarchy may return 0 if no
following take is found

```cpp
void Script::Dialog::UpdateDialogData() {

updates the dialog-data, refresh ResponseTexts, DialogTexts and ResponseIndex

Updates the response and dialog texts for the current state of the dialog.

bool Script::Dialog::AssertDialog() {

executes the assert function from the take's of the current dialog

Executes the assert functions of the takes of the current dialog. Additionally it collects errors in the info log object and in case of errors returns false instead of closing the application.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Script:: DialogDesc
#include <dialogdesc.h>

Inheritance diagram for Script::DialogDesc:

```
Core::RefCounted

Script::DialogDesc
```

Script::DialogDesc Class Reference
Detailed Description

Highlevel description of a dialog. This is one entry in the dialog table created by the dialog managers.

(C) 2006 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DialogDesc ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>void SetGuid (const Util::Guid &amp;guid)</code></td>
<td>Set dialog guid</td>
</tr>
<tr>
<td><code>const Util::Guid &amp; GetGuid () const</code></td>
<td>Get dialog guid</td>
</tr>
<tr>
<td><code>void SetId (const Util::String &amp;id)</code></td>
<td>Set dialog id</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetId () const</code></td>
<td>Get dialog id</td>
</tr>
<tr>
<td><code>void SetSpeaker (const Util::String &amp;speaker)</code></td>
<td>Set dialog speaker</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetSpeaker () const</code></td>
<td>Get dialog speaker</td>
</tr>
<tr>
<td><code>void SetGroup (const Util::String &amp;group)</code></td>
<td>Set dialog speaker group</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetGroup () const</code></td>
<td>Get dialog speaker group</td>
</tr>
<tr>
<td><code>void SetLocked (bool b)</code></td>
<td>Set dialog locked state</td>
</tr>
<tr>
<td><code>bool IsLocked () const</code></td>
<td>Get dialog locked state</td>
</tr>
<tr>
<td><code>void SetConversation (bool b)</code></td>
<td>Set conversation flag</td>
</tr>
<tr>
<td><code>bool IsConversation () const</code></td>
<td>Is conversation</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>Is instance of type Rtti</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf (const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA (const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA (const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetClassName ()</code> const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC <code>GetClassFourCC ()</code> const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

```c
static void DumpRefCountingLeaks ()
   dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
```
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Script::DialogManager
Script::DialogManager Class Reference

#include <dialogmanager.h>

Inheritance diagram for Script::DialogManager:
Detailed Description

**Base** funktions: TalkTo: load a dialog from DB, for a Speaker, and start it. CancelCurrentDialog: Cancel a currently running **Dialog**

Manage Locked an Unlocked States of dialog, and allowes status request Manage if a **Dialog** Take was spoken, an save this data.

Over this Manager the flow of the current active dialog is controlled

A complete overview of the dialog system can be found here: [http://gambar/wiki/index.php/DSA_Story_Subsystem](http://gambar/wiki/index.php/DSA_Story_Subsystem)

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DialogManager ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~DialogManager ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>virtual void OnActivate ()</strong></td>
<td>Called when attached from game server</td>
</tr>
<tr>
<td><strong>virtual void OnDeactivate ()</strong></td>
<td>Called when removed from game server</td>
</tr>
<tr>
<td><strong>virtual void OnLoad ()</strong></td>
<td>Called after loading game state</td>
</tr>
<tr>
<td><strong>virtual void OnSave ()</strong></td>
<td>Called before saving game state</td>
</tr>
<tr>
<td><strong>virtual void SetupAcceptedMessages ()</strong></td>
<td>Override to register accepted messages</td>
</tr>
<tr>
<td><strong>virtual void HandleMessage (const <strong>Ptr</strong>: Script::Dialog &gt; &amp;msg)</strong></td>
<td>Handle a single message</td>
</tr>
<tr>
<td><strong>virtual <strong>Ptr</strong>: Script::Dialog &gt; CreateDialog ()</strong></td>
<td>Create a Dialog</td>
</tr>
<tr>
<td><strong>virtual void StartDialog (const Util::String &amp;speaker)</strong></td>
<td>Start a dialog for a given speaker</td>
</tr>
<tr>
<td><strong>bool HasDialogs (const Util::String &amp;speaker, bool asConversation=false) const</strong></td>
<td>Return true if the entity has dialogs assigned</td>
</tr>
<tr>
<td><strong>void TalkTo (const Util::String &amp;speaker)</strong></td>
<td>Initiate a dialog with an NPC</td>
</tr>
<tr>
<td><strong>bool IsDialogActive () const</strong></td>
<td>Return true if a dialog is currently in progress</td>
</tr>
<tr>
<td><strong>bool IsDialogLoaded () const</strong></td>
<td>Return true if a dialog is loaded</td>
</tr>
</tbody>
</table>
| **virtual void CancelCurrentDialog ()** | }
### Dialog Exists by Id

```cpp
bool DialogExistsById (const Util::String &dialogId) const
default return true if a dialog exists by dialog id
```

### Dialog Take Exists by Id

```cpp
bool DialogTakeExistsById (const Util::String &dialogId, const Util::String &takeId) const
return true if a dialog take exists by take id
```

### Get Speaker

```cpp
virtual const Util::String & GetSpeaker () const
return the Speaker name
```

### Get Dialog Id

```cpp
const Util::String & GetDialogId () const
return the dialog id
```

### Get Dialog Text

```cpp
const Util::Array< Util::String > & GetDialogText () const
get the Take-Text of the current dialog node
```

### Get Responses

```cpp
const Util::Array< Util::String > & GetResponses () const
get an array of texts filled with the response-texts of current dialog node
```

### Get Short Response Texts

```cpp
const Util::Array< Util::String > & GetShortResponseTexts () const
get short response text array
```

### Get Take Sound Id

```cpp
const Util::String & GetTakeSoundID () const
get the CueId of the current take
```

### Get Response Sound Ids

```cpp
const Util::Array< Util::String > & GetResponseSoundIDs () const
get a array with the CuelDs of the current responses
```

### Get Take Speaker

```cpp
const Util::String & GetTakeSpeaker () const
get the Speaker of the current take
```

### Get Response Speakers

```cpp
const Util::Array< Util::String > & GetResponseSpeakers () const
get a array with the Speaker's of the current responses
```

### Get Response Passed States

```cpp
const Util::Array< bool > & GetResponsePassedStates () const
get a array with the passed state of the current responses
```

### Choose Response

```cpp
bool ChooseResponse (int responseIndex)
this function says the manager witch response is chosen by the user
```

### Set Dialog Take State

```cpp
void SetDialogTakeState (const Util::String &dialogId, const Util::String &takeId, bool isPassed)
set the state of a dialog take
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool GetDialogTakeState(const Util::String &amp;dialogId, const Util::String &amp;takeId)</code></td>
<td>get the state of a dialog take</td>
</tr>
<tr>
<td><code>void SetDialogLockState(const Util::String &amp;dialogId, bool locked)</code></td>
<td>sets the locked flag of a dialog</td>
</tr>
<tr>
<td><code>bool GetDialogLockState(const Util::String &amp;dialogId)</code></td>
<td>gets the locked flag of a dialog</td>
</tr>
<tr>
<td><code>void SetLocalizeDialog(bool localize)</code></td>
<td>set localize flag</td>
</tr>
<tr>
<td><code>Ptr&lt;Script::Dialog&gt; GetDialog(const Util::String &amp;speakerOrGroupName, bool loadGroupDialog, bool asConversation=false)</code></td>
<td>get a dialog by speaker or group name</td>
</tr>
<tr>
<td><code>Ptr&lt;Script::Dialog&gt; GetDialog(const Util::String &amp;dialogId)</code></td>
<td>get a dialog by id</td>
</tr>
<tr>
<td><code>bool IsActive() const</code></td>
<td>return true if currently active</td>
</tr>
<tr>
<td><code>virtual void OnBeginFrame()</code></td>
<td>called before frame by the game server</td>
</tr>
<tr>
<td><code>virtual void OnFrame()</code></td>
<td>called per-frame by the game server</td>
</tr>
<tr>
<td><code>virtual void OnEndFrame()</code></td>
<td>called after frame by the game server</td>
</tr>
<tr>
<td><code>virtual void OnStart()</code></td>
<td>called by Game::Server::Start() when the world is started</td>
</tr>
<tr>
<td><code>virtual void OnRenderDebug()</code></td>
<td>render a debug visualization</td>
</tr>
<tr>
<td><code>void AttachPort(const Ptr&lt;Port&gt; &amp;port)</code></td>
<td>attach a message port</td>
</tr>
<tr>
<td><code>void RemovePort(const Ptr&lt;Port&gt; &amp;port)</code></td>
<td>remove a message port</td>
</tr>
<tr>
<td><code>bool HasPort(const Ptr&lt;Port&gt; &amp;port)</code></td>
<td>return true if a port exists</td>
</tr>
</tbody>
</table>
void AttachHandler (const Ptr< Handler > &h)

attach a message handler to the port

void RemoveHandler (const Ptr< Handler > &h)

remove a message handler from the port

void RemoveAllHandlers ()

remove all message handler from the port

SizeT GetNumHandlers () const

return number of handlers attached to the port

const Ptr< Handler > & GetHandlerAtIndex (IndexT i) const

get a message handler by index

virtual void Send (const Ptr< Message > &msg)

send a message to the port

const Util::Array< const Id * > & GetAcceptedMessages () const

get the array of accepted messages (sorted)

bool AcceptsMessage (const Id &msgId) const

return true if port accepts this msg

int GetRefCount () const

get the current refcount

void AddRef ()

increment refcount by one

void Release ()

decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const

return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const

return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const

return true if this object is instance of given class by fourcc

bool Isa (const Rtti &rtti) const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LoadDialogTakeStates()</code></td>
<td>load the table DialogTakeState</td>
</tr>
<tr>
<td><code>SaveDialogTakeStates()</code></td>
<td>save the table DialogTakeState</td>
</tr>
<tr>
<td><code>LoadDialogs()</code></td>
<td>load the Dialog table with locked state</td>
</tr>
<tr>
<td><code>SaveDialogs()</code></td>
<td>save the Dialog table with locked state</td>
</tr>
<tr>
<td><code>ValidateDialog(const Util::Guid &amp;guid)</code></td>
<td>make sure dialog by GUID is loaded</td>
</tr>
<tr>
<td><code>PreloadDialogTables()</code></td>
<td>try preload tables through ScriptTemplateManager</td>
</tr>
<tr>
<td><code>ExecuteTakeActions(const Ptr&lt;DialogTake&gt; &amp;take)</code></td>
<td>executes the actions of a given take</td>
</tr>
<tr>
<td><code>PlayEmotes(const Ptr&lt;DialogTake&gt; &amp;take)</code></td>
<td>play the emotes of a given take.. override this</td>
</tr>
<tr>
<td><code>PlaySound(const Ptr&lt;DialogTake&gt; &amp;take, bool isResponse=false)</code></td>
<td>play sound... override this</td>
</tr>
<tr>
<td><code>LoadAssociatedDialogs(const Util::String &amp;speaker, bool loadForGroup=false)</code></td>
<td>load dialogs associated with an Speaker or group</td>
</tr>
<tr>
<td><code>UnloadDialog()</code></td>
<td>unload currently loaded dialogs</td>
</tr>
<tr>
<td><code>ActivateDialog()</code></td>
<td>activate the Dialog, dialog must be loaded before</td>
</tr>
<tr>
<td><code>AssertDialogTakeExists(const Util::String &amp;dialogTakeKey, const Util::String &amp;dialogId, const Util::String &amp;takeId, const Util::String &amp;callerName)</code></td>
<td>assert that a dialog and take id exists</td>
</tr>
<tr>
<td><code>RegisterMessage(const Id &amp;msgId)</code></td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Script::DialogManager::OnDeactivate() [virtual]

called when removed from game server

This method shuts down the dialog manager when it is removed from the game server.

Reimplemented from `Game::Manager`.

```cpp
void Script::DialogManager::StartDialog(const Util::String& speaker) [virtual]

start a dialog for a given speaker

Start a dialog for a given speaker

```cpp
bool Script::DialogManager::HasDialogs(const Util::String& speakerName, const bool asConversation = false) const

return true if the entity has dialogs assigned

This method checks if dialogs are assigned to an NPC at all.

```cpp
void Script::DialogManager::TalkTo(const Util::String& speaker)

initiate a dialog with an NPC

This method starts a dialog with an Speakername. If a previous dialog is in progress it will be canceled and unloaded.

```cpp
bool ( ) const
Script::DialogManager::IsDialogActive

return true if a dialog is currently in progress

This method returns true if a dialog is currently in progress.

    bool Script::DialogManager::IsDialogLoaded ( ) const

return true if a dialog is loaded

This method returns true if a dialog is loaded

    void Script::DialogManager::CancelCurrentDialog ( ) [virtual]

cancel the currently active dialog

This method cancels and unloads the currently active dialog

    bool Script::DialogManager::DialogTakeExistsById ( "Util::String" dialogId, "Util::String" takeId ) const

return true if a dialogtake exists by take id

Return true if a dialogtake exists by take id.

    const "Util::String" & Script::DialogManager::GetSpeaker ( ) const [virtual]

returns the Speaker name

Returns the speaker name of the currently active dialog.

    const "Util::String" & Script::DialogManager::GetDialogId ( ) const

returns the dialog id
Returns the id of the currently active dialog

```cpp
const Util::Array< Util::String > &
Script::DialogManager::GetDialogText() const
```

get the Take-Text of the current dialog node

Get the take texts of the current dialog node.

```cpp
const Util::Array< Util::String > &
Script::DialogManager::GetResponses() const
```

get an array of texts filled with the response-texts of current dialog node

Get the response texts of the current dialog node.

```cpp
bool
Script::DialogManager::ChooseResponse( int responseIndex )
```

this function says the manager witch response is choosed by the user

Tells the dialog manager which response has been choosen by the user.

```cpp
void
Script::DialogManager::SetDialogTakeState( const Util::String dialogId, 
&
const Util::String takeId, 
&
bool isPassed )
```

set the state of a dialog take

Set a take state to passed (when the take has been displayed).

```cpp
bool
Script::DialogManager::GetDialogTakeState( const Util::String dialogId, 
&
const Util::String takeId 
&
)
get the state of a dialog take

Return the passed flag of a dialog take.

```cpp
void Script::DialogManager::SetDialogLockState
    (const Util::String &dialogId, bool locked)
```

sets the locked flag of a dialog

Set the locked flag of a dialog.

```cpp
bool Script::DialogManager::GetDialogLockState
    (const Util::String &dialogId)
```

gets the locked flag of a dialog

Set the locked flag of a dialog.

```cpp
void Script::DialogManager::LoadDialogTakeStates()
```

load the table DialogTakeState

Load dialog take state table from database.

```cpp
void Script::DialogManager::SaveDialogTakeStates()
```

save the table DialogTakeState

Save passed state flags of takes back into database.

```cpp
void Script::DialogManager::LoadDialogs()
```

load the Dialog table with locked state
Load the dialog table from the database.

```cpp
void Script::DialogManager::SaveDialogs() [protected]
```

save the **Dialog** table with locked state

Save dialog status back to database, only the locked flag can change, so only write this back into the database.

```cpp
void Script::DialogManager::ValidateDialog(const Util::Guid guid&) [protected]
```

make sure dialog by GUID is loaded

Make sure a dialog is loaded by GUID.

```cpp
void Script::DialogManager::PreloadDialogTables() [protected]
```

try preload tables through ScriptTemplateManager

Preload tables from db via ScriptTemplateManager

```cpp
void Script::DialogManager::ExecuteTakeActions(const Ptr<DialogTake> take) [protected, virtual]
```

executes the actions of a given take

Executes the actions of a given take

```cpp
void Script::DialogManager::PlayEmotes(const Ptr<DialogTake> take) [protected, virtual]
```

play the emotes of a given take.. override this

play emotes for current speaker

```cpp
void Script::DialogManager::PlaySound(const Ptr<DialogTake> take,
```
bool isResponse = false

play sound... override this

play sound... override this

bool Script::DialogManager::LoadAssociatedDialogs(const Util::String& speakerOrGroupName, bool loadGroupDialog = false)

[protected]

load dialogs associated with an Speaker or group

This private method loads the dialogtree associated with a given speaker or group from the database into the dialog manager. Returns true if at least one dialog has been activated. if loadGroupDialog is false, speaker will be loaded

**Parameters:**

- `speakerOrGroup` name of a speaker or a group
- `loadGroupDialog` if true dialog for given group name will be loaded

**Returns:**

true, a dialog was found in db

void Script::DialogManager::UnloadDialog()

unload currently loaded dialogs

Unload the currently loaded dialog.

bool Script::DialogManager::ActivateDialog()

activate the Dialog, dialog must be loaded before

Activate a currently loaded dialog.
assert that a dialog and take id exists

Make sure that a dialog/take pair exists.

called before frame by the game server

Called before frame, override in subclasses

Reimplemented in **BaseGameFeature::EntityManager**.

called after frame by the game server

Called after frame, override in subclasses

Reimplemented in **BaseGameFeature::EntityManager**.

attach a message port

Attach a new message port.
Parameters:

*port* pointer to a message port object

```cpp
void Messaging::Dispatcher::RemovePort (const Ptr<Port> &port) [inherited]
```

remove a message port

Remove a message port object.

Parameters:

*handler* pointer to message port object to be removed

```cpp
bool Messaging::Dispatcher::HasPort (const Ptr<Port> &port) const [inherited]
```

return true if a port exists

Return true if a port is already attached.

```cpp
void Messaging::Port::AttachHandler (const Ptr<Handler> &h) [inherited]
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler (const Ptr<Handler> &h) [inherited]
```

remove a message handler from the port

Remove a message handler from the port.
Messaging::Port::Send (Message msg) [virtual, inherited]

send a message to the port

Send a message to the port. This will immediately call the 
HandleMessage() method of all attached handlers. If the message 
has been handled by at least one of the handlers, the Handled() flag 
of the message will be set to true.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Script::<code>DialogTake</code>
Script::DialogTake Class Reference

#include <dialogtake.h>

Inheritance diagram for Script::DialogTake:

```
Core::RefCounted
  ↓
Script::DialogTake
```
Detailed Description

FIXME: the following description is no longer accurate!!!

A dialog take is the smallest element of the dialog system. The take contain a id, a speaker-name, a type, a conditionhierarchy, a list of actions, a list of texts and a list of children. The id identifies the Take in the dialoghierarchy and must unique in the dialog. The speaker-name identifies the 'person' witch speaks. The type identifies the dialogtake as StartState, as Take or as Response. The conditionhierarchy is used to rule if the Take is active or not. The actions are executed if the Take is active. The list of texts represents the dialogtext witch the speaker is calling or the responsetext.

A complete overview of the dialog system can be found here:
http://gambar/wiki/index.php/DSA_Story_Subsystem

(C) 2005 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>take types</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DialogTake()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~DialogTake()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>SetGuid(const Util::Guid &amp;guid)</code></td>
<td>set the ID of the take</td>
</tr>
<tr>
<td><code>GetGuid()</code> const</td>
<td>get the ID of the take</td>
</tr>
<tr>
<td><code>SetId(const Util::String &amp;id)</code></td>
<td>set the ID of the take</td>
</tr>
<tr>
<td><code>GetId()</code> const</td>
<td>get the ID of the take</td>
</tr>
<tr>
<td><code>SetSpeaker(const Util::String &amp;speaker)</code></td>
<td>set the name of the speaker</td>
</tr>
<tr>
<td><code>GetSpeaker()</code> const</td>
<td>get the name of the speaker</td>
</tr>
<tr>
<td><code>SetEmote(const Util::String &amp;emote)</code></td>
<td>set the emote string of the take</td>
</tr>
<tr>
<td><code>GetEmote()</code> const</td>
<td>get the emote string of the take</td>
</tr>
<tr>
<td><code>SetType(const Type type)</code></td>
<td>set the type of the Take</td>
</tr>
<tr>
<td><code>GetType()</code> const</td>
<td>get the type of the Take</td>
</tr>
<tr>
<td><code>SetSound(const Util::String &amp;id)</code></td>
<td>set the sound name of the take</td>
</tr>
<tr>
<td><code>GetSound()</code> const</td>
<td>get the sound name of the take</td>
</tr>
<tr>
<td><code>SetShortText()</code></td>
<td>set the short text of the take</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void Util::String &amp;id)</code></td>
<td>set the short text of the take</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetShortText () const</code></td>
<td>get the short text of the take</td>
</tr>
<tr>
<td><code>void SetTimeStamp (Timing::Time t)</code></td>
<td>set timestamp</td>
</tr>
<tr>
<td><code>Timing::Time GetTimeStamp () const</code></td>
<td>get timestamp</td>
</tr>
<tr>
<td><code>void SetActionRef (const Util::Guid &amp;guid)</code></td>
<td>set action reference guid</td>
</tr>
<tr>
<td><code>const Util::Guid &amp; GetActionRef () const</code></td>
<td>get action reference guid</td>
</tr>
<tr>
<td><code>void SetActionBlock (const Util::Guid &amp;guid)</code></td>
<td>set action block guid</td>
</tr>
<tr>
<td><code>const Util::Guid &amp; GetActionBlock () const</code></td>
<td>get action block guid</td>
</tr>
<tr>
<td><code>void AddText (const Util::String &amp;text)</code></td>
<td>add a text to the textlist</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Util::String &gt; &amp; GetTextList ()</code></td>
<td>get the textlist</td>
</tr>
<tr>
<td><code>void AddAction (const Ptr&lt; Actions::Action &gt; &amp;action)</code></td>
<td>add a action to the actionlist</td>
</tr>
<tr>
<td><code>void SetActionList (const Util::Array&lt; Ptr&lt; Actions::Action &gt; &gt; &amp;actions)</code></td>
<td>set the actionlist</td>
</tr>
<tr>
<td><code>const Util::Array&lt; Ptr&lt; Actions::Action &gt; &gt; &amp; GetActionList ()</code></td>
<td>get the actionlist</td>
</tr>
<tr>
<td><code>void AddChildTake (const Ptr&lt; DialogTake &gt; &amp;child)</code></td>
<td>add a child to the childlist</td>
</tr>
</tbody>
</table>
const Util::Array< Ptr< DialogTake >> & GetChildTakes ()
get the child list

void SetCondition (const Ptr< Conditions::Condition > &condition)
set the condition for this take

const Ptr< Conditions::Condition > & GetCondition () const
get the condition for this take

bool HasCondition ()
return if take has a condition

int GetNumChildTakes () const
get the count of children takes

void ExecuteActions ()
executes the actions of this take

bool Assert (const Ptr< Script::InfoLog > &infoLog)
executes the assert functions from the members of this take

int GetRefCount () const
get the current ref count

void AddRef ()
increment ref count by one

void Release ()
decrement ref count and destroy object if ref count is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by four cc

bool
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetClassName</strong> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC <strong>GetClassFourCC</strong> () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td><strong>StringToType</strong></td>
<td>(const Util::String &amp;str) convert string to take type</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong></td>
<td>() dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
bool Script::DialogTake::Assert(const Ptr<Script::InfoLog> infoLog &)
```

executes the assert functions from the members of this take

Executes the assert functions from the members of this take. Additionally it collects errors in the info log object and returns false in case of errors instead of closing the application.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

**Util::FourCC**
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Script::InfoLog
Script::InfoLog Class Reference

#include <infolog.h>

Inheritance diagram for Script::InfoLog:
Detailed Description

To an info log object you can add several strings which can contain different information like debug information or error messages.

You can also specify a log level, a short description and a source (e.g. information on the creator of the info log object).

Additionally you can open and close sections for a better separation of general and detailed information. Within a section you can open new sections, too but there is no check if the amount of openings fits to the amount of closings. For further details see Description of BeginSection() and EndSection().

Per default description, source and the list of information strings are empty. The log level will be 'Non'.

An info log object is based on an array of strings.

(C) 2006 Radon Labs GmbH
Public Types

```c
enum LogLevel {
    Non = 0, Debug = (1<<0), Warning = (1<<1), Error = (1<<2),
    Graphic = (1<<3)
}
```
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InfoLog()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~InfoLog()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>void SetDescription(Util::String description)</code></td>
<td>sets the short description</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetDescription()</code></td>
<td>gets the short description</td>
</tr>
<tr>
<td><code>void SetSource(Util::String source)</code></td>
<td>sets the source of this info log (e.g. creator)</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetSource()</code></td>
<td>gets the source of this info log (e.g. creator)</td>
</tr>
<tr>
<td><code>void SetLogLevel(LogLevel logLvl)</code></td>
<td>sets the log level</td>
</tr>
<tr>
<td><code>LogLevel GetLogLevel()</code></td>
<td>gets the log level</td>
</tr>
<tr>
<td><code>void AddInfo(const Util::String &amp;infoString)</code></td>
<td>adds an information string to this info log</td>
</tr>
<tr>
<td><code>void AddInfo(const Util::Array&lt;Util::String&gt; &amp;infoArray)</code></td>
<td>adds an array of information strings to this info log</td>
</tr>
<tr>
<td><code>const Util::Array&lt;Util::String&gt; &amp; GetInfo()</code></td>
<td>gets the list of information strings of this info log</td>
</tr>
<tr>
<td><code>void BeginSection(const Util::String &amp;beginInfo)</code></td>
<td>starts new section</td>
</tr>
<tr>
<td><code>void EndSection(const Util::String &amp;endInfo)</code></td>
<td>ends current section</td>
</tr>
<tr>
<td><code>bool HasInfo()</code></td>
<td>returns true if information has been added</td>
</tr>
<tr>
<td><code>bool HasInfoInSection()</code></td>
<td>returns true if information has been added to current</td>
</tr>
<tr>
<td>in</td>
<td>description</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><code>Util::String</code> <strong>ToString</strong> () const</td>
<td>returns a simple string representation of this info log</td>
</tr>
<tr>
<td>int <code>GetRefCount</code> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef</code> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release</code> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Util::String</code> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Util::FourCC</code> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Util::String</code> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp; <code>GetClassName</code> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC</code> <code>GetClassFourCC</code> () const</td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>Function</td>
<td>Parameters</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td><code>LogLevelToString</code></td>
<td><code>LogLevel logLvl</code></td>
</tr>
<tr>
<td><code>StringToLogLevel</code></td>
<td><code>const Util::String &amp;logLevelString</code></td>
</tr>
<tr>
<td><code>DumpRefCountingLeaks</code></td>
<td></td>
</tr>
</tbody>
</table>
### Static Public Attributes

<table>
<thead>
<tr>
<th>static const Util::String</th>
<th>BEGINLABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>label which is used to mark begin section tag</em></td>
</tr>
<tr>
<td>static const Util::String</td>
<td>ENDLABEL</td>
</tr>
<tr>
<td></td>
<td><em>label which is used to mark end section tag</em></td>
</tr>
</tbody>
</table>
Member Enumeration Documentation

enum Script::InfoLog::LogLevel

A log level which specifies type or/and origin of the info log object. It's possible to combine different log levels by using bit operators.

**Enumerator:**

- **Non**  no log level (general information)
- **Debug** debug information
- **Warning** warning information
- **Error**  error information
- **Graphic** information out of a special domain
Constructor & Destructor Documentation

Script::InfoLog::InfoLog ( )

constructor

constructor

Script::InfoLog::~InfoLog ( ) [virtual]

destructor

destructor
Member Function Documentation

void
Script::InfoLog::SetDescription (Util::String description)

sets the short description
Sets the description.

const Util::String &
Script::InfoLog::GetDescription ( ) const

gets the short description
Gets the description.

void
Script::InfoLog::SetSource (Util::String source)

sets the source of this info log (e.g. creator)
Sets the source of this info log (e.g. creator).

const Util::String &
Script::InfoLog::GetSource ( ) const

gets the source of this info log (e.g. creator)
Gets the source of this info log (e.g. creator).

void
Script::InfoLog::SetLogLevel (LogLevel logLvl)

sets the log level
Sets the log level. There will be no check if the log level is supported.

InfoLog::LogLevel
Script::InfoLog::GetLogLevel ( ) const

gets the log level
Gets the log level.

```cpp
void Script::InfoLog::AddInfo(const Util::String infoString &)
```

adds an information string to this info log

Adds an information string to this info log.

```cpp
void Script::InfoLog::AddInfo(const Util::Array<Util::String> & infoArray )
```

adds an array of information strings to this info log

Adds an array of information strings to this info log.

```cpp
const Util::Array<Util::String> & Script::InfoLog::GetInfo()
```

gets the list of information strings of this info log

Gets a list of information strings.

```cpp
void Script::InfoLog::BeginSection(const Util::String beginInfo &)
```

starts new section

Start new info log section.

The begin tags for open sections will be added to the list of information strings when the first information in this section has been added.

The begin section tag will have following format: "<InfoLog::BEGINLABEL><beginInfo>".

See `EndSection()`.

See `InfoLog::BEGINLABEL`. 
void Script::InfoLog::EndSection (const Util::String endInfo )

ends current section

End current info log section.

If there is a begin section tag without information the begin and the end section tags won’t be added to the info log object. If there is a begin section tag with following information the end section tag will be added to the info log object.

The end section tag will always end the last open section tag, therefor to every begin section tag an end section tag should be placed. This method does not check if the amount of begin tags fits to the amount of end tags!

The end section tag will have following format: 
"<InfoLog::ENDLABEL><endInfo>"

See BeginSection(). See InfoLog::ENDLABEL.

bool Script::InfoLog::HasInfo ( ) const

returns true if information has been added

Returns true if information has been added.

bool Script::InfoLog::HasInfoInCurrentSection ( ) const

returns true if information has been added to current section

Returns true if information has been added to current section.

Util::String Script::InfoLog::ToString ( ) const

returns a simple string representation of this info log
Returns an simple string representation of the information strings. The begin and end labels will be removed and instead the inner sections will be indented.

```cpp
Util::String
Script::InfoLog::LogLevelToString ( InfoLog::LogLevel logLvl ) [static]
```

converts a log level to a string

Converts a log level to a string.

If there are set more than one bit of the log level the names of the log levels will be seperated by " | " . The string will be empty if no set bit of the log level correspond to a supported log level.

```cpp
InfoLog::LogLevel
Script::InfoLog::StringToLogLevel ( const Util::String & logLevelString ) [static]
```

converts a string to a log level

Converts a string to a log level.

This method uses a simple pattern matching. If the name of a log level is in the log level string the corresponding bit will be set.

It's not case sensitiv. Not supported log level names will be ignored. If there are no supported log levels the method returns log level "Non".

```cpp
int
Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void
Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

Script::InfoLogFilter
Script::InfoLogFilter Class Reference

#include <infologfilter.h>

Inheritance diagram for Script::InfoLogFilter:

```
Core::RefCounted

Script::InfoLogFilter
```

Detailed Description

Provides methods to filter arrays of info log objects by the info log's description, source, log level or/and information strings.

Additionally it provides methods to check if an info log object contains a string pattern in it's description, source or information strings or if it's log level fits to log level mask.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td><strong>GetRefCount</strong> () const</td>
<td>gets the current refcount</td>
</tr>
<tr>
<td>void</td>
<td><strong>AddRef</strong> ()</td>
<td>increments refcount by one</td>
</tr>
<tr>
<td>void</td>
<td><strong>Release</strong> ()</td>
<td>decrements refcount and destroys object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
<td>returns true if this object is an instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
<td>returns true if this object is an instance of given class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
<td>returns true if this object is an instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
<td>returns true if this object is an instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
<td>returns true if this object is an instance of given class, or a derived class by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
<td>returns true if this object is an instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
<td>gets the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
<td>gets the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| `static bool ContainsDescription (Ptr<
    Script::InfoLog > il, const
    Util::String &dPattern)`                  | Returns all info logs which contain specified description pattern |
| `static bool ContainsSource (Ptr<
    Script::InfoLog > il, const
    Util::String &sPattern)`                  | Returns all info logs which contain specified source pattern |
| `static bool ContainsLogLevel (Ptr<
    Script::InfoLog > il,
    Script::InfoLog::LogLevel
    logLvlMask)`                               | Returns all info logs which fit to the specified log level mask |
| `static bool ContainsInformation (Ptr<
    Script::InfoLog > il, const
    Util::String &iPattern)`                  | Returns all info logs which contain specified information pattern |
| `static Util::Array< Ptr< Script::InfoLog > >
  FilterByDescription (const
  Util::Array< Ptr< Script::InfoLog
  > > &ils, const Util::String
  &dPattern)`                                | Returns all info logs which contain specified description pattern |
| `static Util::Array< Ptr< Script::InfoLog > >
  FilterBySource (const
  Util::Array< Ptr< Script::InfoLog
  > > &ils, const Util::String
  &sPattern)`                                | Returns all info logs which contain specified source pattern |
| `static Util::Array< Ptr< Script::InfoLog > >
  FilterByLogLevel (const
  Util::Array< Ptr< Script::InfoLog
  > > &ils,
  Script::InfoLog::LogLevel
  logLvlMask)`                                | Returns all info logs which fit to the specified log level mask |
| `static Util::Array< Ptr< Script::InfoLog > >
  FilterByInformation (const
  Util::Array< Ptr< Script::InfoLog
  > > &ils, const Util::String)`             | Returns all info logs which contain specified information pattern |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter (const Util::Array&lt; Ptr<a href="">Script::InfoLog</a> &gt; &amp;ils, const Util::String &amp;dPattern, const Util::String &amp;sPattern, Script::InfoLog::LogLevel logLvlMask, const Util::String &amp;iPattern)</td>
<td>returns all info logs which contain specified information pattern</td>
</tr>
<tr>
<td>static Util::Array&lt; Ptr<a href="">Script::InfoLog</a> &gt; &gt;</td>
<td></td>
</tr>
</tbody>
</table>
| static void DumpRefCountingLeaks () | dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

```cpp
static void DumpRefCountingLeaks ()
```
Member Function Documentation

bool Script::InfoLogFilter::ContainsDescription (Ptr<Script::InfoLog> il, const Util::String & dPattern)
returns true if description of the info log contains specified string pattern

bool Script::InfoLogFilter::ContainsSource (Ptr<Script::InfoLog> il, const Util::String & sPattern)
returns true if source of the info log contains specified string pattern

bool Script::InfoLogFilter::ContainsLogLevel (Ptr<Script::InfoLog> il, Script::InfoLog::LogLevel logLvlMask)
returns true if log level of the info log fits to the specified log level mask

bool Script::InfoLogFilter::ContainsInformation (Ptr<Script::InfoLog> il, const Util::String & iPattern)
returns true if information of the info log contains specified string pattern

Util::Array<Ptr<Script::InfoLog>> Script::InfoLogFilter::FilterByDescription (Ptr<Util::Array<Ptr<Script::InfoLog>>> ils,

returns true if description of the info log contains specified string pattern
returns all info logs which contain specified description pattern

Returns all info logs which contain specified description pattern.

returns all info logs which contain specified source pattern

Returns all info logs which contain specified source pattern.

returns all info logs which fit to the specified log level mask

Returns all info logs which fit to the specified log level mask.

returns all info logs which contain specified information pattern

Returns all info logs which contain specified information pattern.
returns all info logs which contain specified description, source and information pattern and fit to the specified log level mask

```
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Script::ScriptManager
Script::ScriptManager Class Reference

#include <scriptmanager.h>

Inheritance diagram for Script::ScriptManager:

```
   Core::RefCounted
    ↑                  ↓
    ↓                  ↓
  Messaging::Port       Messaging::Dispatcher
    ↑                  ↑
    ↓                  ↓
  Game::Manager
    ↑
  Script::ScriptManager
```
Detailed Description

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ScriptManager ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~ScriptManager ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>OnActivate ()</strong></td>
<td>Called when attached to game server</td>
</tr>
<tr>
<td><strong>OnDeactivate ()</strong></td>
<td>Called when removed from game server</td>
</tr>
<tr>
<td><strong>LoadConditionScripts</strong> (const</td>
<td>Load a list of condition scripts and build a condition block</td>
</tr>
<tr>
<td>Util::Array&lt; Util::String &gt; &amp;scripts, bool combineWithOr=false, const <strong>Ptr&lt;</strong></td>
<td>**Script::InfoLog &gt; &amp;infoLog=0)</td>
</tr>
<tr>
<td><strong>LoadActionScripts</strong> (const Util::Array&lt; Util::String &gt; &amp;scripts, const <strong>Ptr&lt;</strong></td>
<td>**Script::InfoLog &gt; &amp;infoLog=0)</td>
</tr>
<tr>
<td><strong>LoadStatement</strong> (const Util::Guid &amp;blockGuid, const Util::Guid &amp;startId, const Util::String &amp;table=&quot;_Scripts_Statements&quot;)</td>
<td>Load a statement block</td>
</tr>
<tr>
<td><strong>LoadCondition</strong> (const Util::Guid &amp;blockGuid, const Util::Guid &amp;startId, const Util::String &amp;table=&quot;_Scripts_Conditions&quot;)</td>
<td>Load condition</td>
</tr>
<tr>
<td><strong>HasActionScript</strong> (const Util::String &amp;script)</td>
<td>Check if action script is available</td>
</tr>
<tr>
<td><strong>LoadActionFromScript</strong> (const Util::String &amp;scriptName, const <strong>Ptr&lt;</strong></td>
<td>Load action from script</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Script::InfoLog &gt; &amp;infoLog=0)</td>
<td>Load an action from a script (sequence action if needed)</td>
</tr>
<tr>
<td>LoadStateMachine (const Util::String &amp;stateMachineName, const Ptr<a href="">Game::Entity</a> &amp;entity)</td>
<td>Load state machine</td>
</tr>
<tr>
<td>virtual void ReloadScripts ()</td>
<td>Debug helper function to re-initialize scripts</td>
</tr>
<tr>
<td>bool IsActive () const</td>
<td>Return true if currently active</td>
</tr>
<tr>
<td>virtual void OnBeginFrame ()</td>
<td>Called before frame by the game server</td>
</tr>
<tr>
<td>virtual void OnFrame ()</td>
<td>Called per-frame by the game server</td>
</tr>
<tr>
<td>virtual void OnEndFrame ()</td>
<td>Called after frame by the game server</td>
</tr>
<tr>
<td>virtual void OnLoad ()</td>
<td>Called after loading game state</td>
</tr>
<tr>
<td>virtual void OnSave ()</td>
<td>Called before saving game state</td>
</tr>
<tr>
<td>virtual void OnStart ()</td>
<td>Called by Game::Server::Start() when the world is started</td>
</tr>
<tr>
<td>virtual void OnRenderDebug ()</td>
<td>Render a debug visualization</td>
</tr>
<tr>
<td>virtual void HandleMessage (const Ptr<a href="">Messaging::Message</a> &amp;msg)</td>
<td>Handle a single message (distribute to ports which accept the message)</td>
</tr>
<tr>
<td>void AttachPort (const Ptr&lt;Port&gt; &amp;port)</td>
<td>Attach a message port</td>
</tr>
<tr>
<td>void RemovePort (const Ptr&lt;Port&gt; &amp;port)</td>
<td>Remove a message port</td>
</tr>
<tr>
<td>bool HasPort (const Ptr&lt;Port&gt; &amp;port) const</td>
<td>Return true if a port exists</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void SetupAcceptedMessages()</td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>void AttachHandler(const Ptr&lt;Handler&gt; &amp;h)</td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td>void RemoveHandler(const Ptr&lt;Handler&gt; &amp;h)</td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td>void RemoveAllHandlers()</td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td>SizeT GetNumHandlers() const</td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td>const Ptr&lt;Handler&gt; &amp; GetHandlerAtIndex(IndexT i) const</td>
<td>get a message handler by index</td>
</tr>
<tr>
<td>virtual void Send(const Ptr&lt;Message&gt; &amp;msg)</td>
<td>send a message to the port</td>
</tr>
<tr>
<td>const Util::Array&lt;const Id *&gt; &amp; GetAcceptedMessages() const</td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td>bool AcceptsMessage(const Id &amp;msgId) const</td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

void **LoadCanScript** ()

*begin a loading sequenz for can scripts, keep dbreader open*

void **UnloadCanScript** ()

*end loading sequenz for can scripts, clear dbreader*

void **LoadOnScript** ()

*begin a loading sequenz for on scripts, keep dbreader open*

void **UnloadOnScript** ()

*end loading sequenz for on scripts, clear dbreader*

void **LoadStateMachines** ()

*begin a loading sequenz for can scripts, keep dbreader open*

void **UnloadStateMachines** ()

*end loading sequenz for can scripts, clear dbreader*

void **BuildStateMachineRacks** ()

*build stateMachineRacks dictionary*

int **FindGuidIndex** (const Util::Dictionary< Util::Guid, int >&statementListIndices, const Util::Guid &guid)

*helper function to find statement with specific guid, return -1 on failure*

void **LoadTableContent** (const Util::Guid &blockGuid, const Util::Guid &startId, const Util::String &table, Util::Array< Statement > &statementList, Util::Dictionary< Util::Guid, int > &statementListIndices)

*load content from a table*

void **FillActionAndConditionLists** (Util::Array< Statement > &statementList, Util::Dictionary< Util::Guid, int > &statementListIndices)

*fill action and condition list*

void **ConstructContent** (Util::Array< Statement > &statementList, Util::Dictionary< Util::Guid, int > &statementListIndices)

*fill action and conditions with content*

void **LoadStates** (Ptr< FSM::StateMachine >)
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadTransitions (Ptr&lt; FSM::State &gt; state, const Util::String machineName, StateRack indexRack)</td>
<td>load transitions for a state</td>
</tr>
<tr>
<td>LoadState (const Util::String &amp;stateMachineName, StateRack rack, IndexT indexRack)</td>
<td>load a statemachine State</td>
</tr>
<tr>
<td>LoadTransition ()</td>
<td>load a transition</td>
</tr>
<tr>
<td>RegisterMessage (const Id &amp;msgId)</td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

\texttt{Ptr< Actions::ActionList > Script::ScriptManager::LoadStatement} (const \texttt{Util::Guid} & \texttt{blockGuid},
const \texttt{Util::Guid} & \texttt{startId},
const \texttt{Util::String} & \texttt{table} = "_Scripts_Statements"
)

load a statement block

Load statement block from db

\texttt{Ptr< Conditions::Condition > Script::ScriptManager::LoadCondition} (const \texttt{Util::Guid} & \texttt{blockGuid},
const \texttt{Util::Guid} & \texttt{startId},
const \texttt{Util::String} & \texttt{table} = "_Scripts_Conditions"
)

load condition

Load condition from db

\texttt{Ptr< Actions::Action > Script::ScriptManager::LoadActionFromScript} (const \texttt{Util::String} & \texttt{scriptName},
\texttt{const Ptr< Script::InfoLog > infoLog = @0}
)

load an action from a script (sequence action if needed)

Load an action from a given script name the action is allways a sequence action
void Game::Manager::OnBeginFrame() [virtual, inherited]
called before frame by the game server
Called before frame, override in subclasses
Reimplemented in BaseGameFeature::EntityManager.

void Game::Manager::OnEndFrame() [virtual, inherited]
called after frame by the game server
Called after frame, override in subclasses
Reimplemented in BaseGameFeature::EntityManager.

void Messaging::Dispatcher::HandleMessage(const Ptr<Messaging::Message> &msg) [virtual, inherited]
handle a single message (distribute to ports which accept the message)
Handle a message. The message will only be distributed to ports which accept the message.
Reimplemented from Messaging::Port.
Reimplemented in Script::DialogManager.

void Messaging::Dispatcher::AttachPort(const Ptr<Port> &port) [inherited]
attach a message port
Attach a new message port.

Parameters:

  port  pointer to a message port object
void Messaging::Dispatcher::RemovePort(const Ptr<Port> &port) [inherited]

remove a message port

Remove a message port object.

**Parameters:**

- `handler` pointer to message port object to be removed

bool Messaging::Dispatcher::HasPort(const Ptr<Port> &port) const [inherited]

return true if a port exists

Return true if a port is already attached.

void Messaging::Port::AttachHandler(const Ptr<Handler> &h) [inherited]

attach a message handler to the port

Attach a message handler to the port.

void Messaging::Port::RemoveHandler(const Ptr<Handler> &h) [inherited]

remove a message handler from the port

Remove a message handler from the port.

void Messaging::Port::Send(const Ptr<Message> &msg) [virtual, inherited]
Send a message to the port. This will immediately call the
HandleMessage() method of all attached handlers. If the message
has been handled by at least one of the handlers, the Handled() flag
of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Script::SubstitutionManager
Script::SubstitutionManager Class Reference

#include <substitutionmanager.h>

Inheritance diagram for Script::SubstitutionManager:
Detailed Description

replaces a text string token, depending on the current selection leader

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SubstitutionManager ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~SubstitutionManager ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Util::String GetSubstitution (const Util::String &amp;s, Ptr<a href="">Game::Entity</a> entity)</strong></td>
<td>get the substituted string by a given string and corresponding entity</td>
</tr>
<tr>
<td>virtual void <strong>OnActivate ()</strong></td>
<td>called when attached to game server</td>
</tr>
<tr>
<td>virtual void <strong>OnDeactivate ()</strong></td>
<td>called when removed from game server</td>
</tr>
<tr>
<td>bool <strong>IsActive () const</strong></td>
<td>return true if currently active</td>
</tr>
<tr>
<td>virtual void <strong>OnBeginFrame ()</strong></td>
<td>called before frame by the game server</td>
</tr>
<tr>
<td>virtual void <strong>OnFrame ()</strong></td>
<td>called per-frame by the game server</td>
</tr>
<tr>
<td>virtual void <strong>OnEndFrame ()</strong></td>
<td>called after frame by the game server</td>
</tr>
<tr>
<td>virtual void <strong>OnLoad ()</strong></td>
<td>called after loading game state</td>
</tr>
<tr>
<td>virtual void <strong>OnSave ()</strong></td>
<td>called before saving game state</td>
</tr>
<tr>
<td>virtual void <strong>OnStart ()</strong></td>
<td>called by Game::Server::Start() when the world is started</td>
</tr>
<tr>
<td>virtual void <strong>OnRenderDebug ()</strong></td>
<td>render a debug visualization</td>
</tr>
<tr>
<td>virtual void <strong>HandleMessage (const Ptr<a href="">Messaging::Message</a> &amp;msg)</strong></td>
<td>handle a single message (distribute to ports which accept the message)</td>
</tr>
<tr>
<td>void <strong>AttachPort (const Ptr&lt;Port&gt; &amp;port)</strong></td>
<td>attach a message port</td>
</tr>
</tbody>
</table>
void **RemovePort** (const **Ptr**< **Port** > &port)  
remove a message port

bool **HasPort** (const **Ptr**< **Port** > &port) const  
return true if a port exists

virtual void **SetupAcceptedMessages** ()  
override to register accepted messages

void **AttachHandler** (const **Ptr**< **Handler** > &h)  
attach a message handler to the port

void **RemoveHandler** (const **Ptr**< **Handler** > &h)  
remove a message handler from the port

void **RemoveAllHandlers** ()  
remove all message handler from the port

**SizeT** **GetNumHandlers** () const  
return number of handlers attached to the port

const **Ptr**< **Handler** > & **GetHandlerAtIndex** (IndexT i) const  
get a message handler by index

virtual void **Send** (const **Ptr**< **Message** > &msg)  
send a message to the port

const **Util::Array**< **const** Id * > & **GetAcceptedMessages** () const  
get the array of accepted messages (sorted)

bool **AcceptsMessage** (const Id &msgId) const  
return true if port accepts this msg

int **GetRefCount** () const  
get the current refcount

void **AddRef** ()  
increment refcount by one

void **Release** ()  
decrement refcount and destroy object if refcount is zero

bool **IsInstanceOf** (const Rtti &rtti) const  
return true if this object is instance of given class

bool **IsInstanceOf** (const **Util::String** &className) const  
return true if this object is instance of given class by string

bool **IsInstanceOf** (const **Util::FourCC** &classFourCC) const
<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Rtti &amp;rtti) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::String &amp;rttiName) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC</th>
<th>GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

`void RegisterMessage (const Id &msgId)`

*register a single accepted message*
Member Function Documentation

```cpp
const Util::String &

Script::SubstitutionManager::GetSubstitution(const Util::String & s, 
    Ptr<Game::Entity> entity)
```

get the substituted string by a given string and corresponding entity

override to get localized string by token and entity

```cpp
void Game::Manager::OnActivate()
```

called when attached to game server

This method is called when the manager is attached to the game server. The manager base class will register its message port with the message server.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager, BaseGameFeature::EnvQueryManager, BaseGameFeature::GlobalAttrsManager, Script::DialogManager, and Script::ScriptManager.

```cpp
void Game::Manager::OnDeactivate()
```

called when removed from game server

This method is called when the manager is removed from the game server. It will unregister its message port from the message server at this point.

Reimplemented in BaseGameFeature::TimeManager, BaseGameFeature::CategoryManager,
BaseGameFeature::EntityManager, BaseGameFeature::EnvEntityManager, BaseGameFeature::EnvQueryManager, Script::DialogManager, and Script::ScriptManager.

void Game::Manager::OnBeginFrame() [virtual, inherited]
called before frame by the game server
Called before frame, override in subclasses
Reimplemented in BaseGameFeature::EntityManager.

void Game::Manager::OnEndFrame() [virtual, inherited]
called after frame by the game server
Called after frame, override in subclasses
Reimplemented in BaseGameFeature::EntityManager.

void Messaging::Dispatcher::HandleMessage(const Ptr<Messaging::Message>& msg) [virtual, inherited]
handle a single message (distribute to ports which accept the message)
Handle a message. The message will only be distributed to ports which accept the message.
Reimplemented from Messaging::Port.
Reimplemented in Script::DialogManager.

void Messaging::Dispatcher::AttachPort(const Ptr<Port>& port) [inherited]
attach a message port

Attach a new message port.

**Parameters:**

*port* pointer to a message port object

```cpp
void Messaging::Dispatcher::RemovePort(const Ptr<Port>& port) [inherited]
```

remove a message port

Remove a message port object.

**Parameters:**

*handler* pointer to message port object to be removed

```cpp
bool Messaging::Dispatcher::HasPort(const Ptr<Port>& port) const [inherited]
```

return true if a port exists

Return true if a port is already attached.

```cpp
void Messaging::Port::AttachHandler(const Ptr<Handler>& h) [inherited]
```

attach a message handler to the port

Attach a message handler to the port.

```cpp
void Messaging::Port::RemoveHandler(const Ptr<Handler>& h) [inherited]
```

remove a message handler from the port
Remove a message handler from the port.

```cpp
void Messaging::Port::Send(const Ptr<Message> &msg) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the **HandleMessage()** method of all attached handlers. If the message has been handled by at least one of the handlers, the Handled() flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC () const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Script::Task
Script::Task Class Reference

#include <task.h>

Inheritance diagram for Script::Task:

Core::RefCounted
   
Script::Task
Detailed Description

Represents a task. All tasks without a parentId will be loaded into memory, but the sub tasks will only be loaded for active (unlocked) tasks.

(C) 2006 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>State</th>
<th>quest states</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><strong>AssertMode</strong></td>
<td><em>assert mode for assert method</em></td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Task ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~Task ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Load (const Ptr&lt; Db::Reader &gt; &amp;dbReader)</code></td>
<td>Initialize quest from database</td>
</tr>
<tr>
<td><code>Save (const Ptr&lt; Db::Writer &gt; &amp;dbWriter, const Util::String &amp;parentName=&quot;&quot;)</code></td>
<td>Save the quest back to the database</td>
</tr>
<tr>
<td><code>IsLoaded () const</code></td>
<td>Return true if quest is loaded</td>
</tr>
<tr>
<td><code>LoadSubTasks ()</code></td>
<td>Load the quest tasks of this quest from the database</td>
</tr>
<tr>
<td><code>HasOpenConditions () const</code></td>
<td>Return true if task has open conditions</td>
</tr>
<tr>
<td><code>GetOpenConditions () const</code></td>
<td>Get the open conditions</td>
</tr>
<tr>
<td><code>GetCloseConditions () const</code></td>
<td>Get the close conditions</td>
</tr>
<tr>
<td><code>GetFailConditions () const</code></td>
<td>Get fail conditions</td>
</tr>
<tr>
<td><code>LoadOpenConditions ()</code></td>
<td>Load open conditions</td>
</tr>
<tr>
<td><code>LoadCloseConditions ()</code></td>
<td>Load close conditions</td>
</tr>
<tr>
<td><code>LoadFailConditions ()</code></td>
<td>Load fail conditions</td>
</tr>
<tr>
<td><code>LoadOpenActions ()</code></td>
<td>Load open actions</td>
</tr>
<tr>
<td><code>LoadCloseActions ()</code></td>
<td>Load close actions</td>
</tr>
</tbody>
</table>
void LoadFailActions ()
load fail actions

void UnloadSubTasks ()
unload the quest tasks of this quest

bool AreSubTasksLoaded () const
return true if tasks are loaded

bool Assert (AssertMode assertMode)
asserts open or close conditions or open or close actions

bool EvaluateOpenConditions ()
evaluate the quest task open conditions

bool EvaluateCloseConditions ()
evaluate the quest task closed conditions

bool EvaluateFailConditions ()
evaluate fail conditions

void ExecuteOpenActions ()
execute the open actions

void ExecuteCloseActions ()
execute the close actions

void ExecuteFailActions ()
execute fail actions

const Util::Guid & GetGuid () const
get the quest's guid

const Util::Guid & GetParentGuid () const
get parent guid

State GetState () const
get the current quest state

const Util::String & GetId () const
get the quest id string

const Util::String & GetTitle () const
get the title

const Util::Array< Ptr< Task > > & GetSubTasks () const
get the tasks of the quest

int GetNumSubTasks () const
get number of subtasks

bool HasSubTasks () const
has subtasks
void SetOpenText (const Util::String &t)  
set open text

const Util::String & GetOpenText () const  
get open text

void SetCloseText (const Util::String &t)  
set close text

const Util::String & GetCloseText () const  
get close text

void SetFailedText (const Util::String &t)  
set failed text

const Util::String & GetFailedText () const  
get failed text

Util::String GetText () const  
get current state text

void UnlockTask ()  
unlock the task

void CloseTask ()  
close the task

void FailTask ()  
set the task failed

void EvaluateSubTasks ()  
evaluate the subtasks

bool HasTaskStatusChanged () const  
checks if task status has changed

void SetTaskStatusUnchanged ()  
set task status on unchanged (for updating GUI)

const Util::String & GetTargetEntityName () const  
set task target entity

Ptr< Script::Task > FindSubTaskByGuid (const Util::Guid &guid)  
finds a single task by guid

Ptr< Script::Task > FindSubTaskById (const Util::String &Id)  
finds a single task by Id

void SetParentKey (const Util::String &key)
```cpp
set parent key

const Util::String & GetParentKey () const
get parent key

void SetTaskViewed (bool b)
set task viewed

bool GetTaskViewed () const
get task viewed state

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
get the class name

Util::FourCC GetClassFourCC () const
get the class FourCC code
```
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static State StringToString (const Util::String &amp;s)</td>
<td>convert string to state</td>
</tr>
<tr>
<td>static Util::String StateToString (State s)</td>
<td>convert state to string</td>
</tr>
<tr>
<td>static void DumpRefCountingLeaks ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Script::Task::Load(
    const Ptr<Db::Reader> dbReader
) { initialize quest from database }

This loads the quest from the database.

```cpp
void Script::Task::Save(
    const Ptr<Db::Writer> dbWriter,
    const Util::String& parentName ="
) { save the quest back to the database }

Save quest status back to database.

```cpp
void Script::Task::LoadSubTasks() { load the quest tasks of this quest from the database }

Loads all tasks of the quest. This method is usually called automatically as soon as a SetState(Unlocked) is invoked.

```cpp
void Script::Task::UnloadSubTasks() { unload the quest tasks of this quest }

Unloads all tasks of the quest. This method is usually called automatically as soon as a SetState(Closed) is invoked.

```cpp
bool Script::Task::EvaluateOpenConditions() { }
```
evaluate the quest task open conditions

Evaluate open conditions.

```cpp
bool
Script::Task::EvaluateCloseConditions()
```

evaluate the quest task closed conditions

This evaluates the closed conditions of all open quest tasks. Should only be called when the quest is in the unlocked state. If the ClosedConditions of a task are true, the task will be closed.

```cpp
bool
Script::Task::EvaluateFailConditions()
```

evaluate fail conditions

This evaluates the closed conditions of all open quest tasks. Should only be called when the quest is in the unlocked state. If the ClosedConditions of a task are true, the task will be closed.

```cpp
int
Script::Task::GetNumSubTasks() const
```

get number of subtasks

get the parent guid string

```cpp
bool
Script::Task::HasSubTasks() const
```

has subtasks

has subtasks

```cpp
void
Script::Task::CloseTask()
```

close the task

This is the public method to close the quest. If the quest is not
currently in the unlocked state, nothing will happen

```c
void
Script::Task::FailTask();
```

set the task failed

This is the public method to set the task failed. If the task is not currently in the unlocked state, nothing will happen

```c
void
Script::Task::EvaluateSubTasks();
```

evaluate the subtasks

evaluate open conditions of the subtasks, or if they're already open evaluate their subtasks

```c
Ptr< Script::Task >
Script::Task::FindSubTaskByGuid(const Util::Guid guid &);
```

finds a single task by guid

```c
Ptr< Script::Task >
Script::Task::FindSubTaskById(const Util::String id &);
```

finds a single task by Id

```c
int
Core::RefCounted::GetRefCount(const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```c
void
Core::RefCounted::AddRef([inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
ScriptFeature::ScriptFeatureUnit
ScriptFeature::ScriptFeatureUnit Class Reference

#include <scriptfeatureunit.h>

Inheritance diagram for ScriptFeature::ScriptFeatureUnit:

```
Core::RefCounted
    |
    v
Game::FeatureUnit
    |
    v
ScriptFeature::ScriptFeatureUnit
```
Detailed Description

The **ScriptFeatureUnit** delivers a handful servers for successfully loading, playing and controlling of scripts (also cutscenes)

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ScriptFeatureUnit()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~ScriptFeatureUnit()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <code>OnActivate()</code></td>
<td>Called from <code>GameServer::ActivateProperties()</code></td>
</tr>
<tr>
<td>virtual void <code>OnDeactivate()</code></td>
<td>Called from <code>GameServer::DeactivateProperties()</code></td>
</tr>
<tr>
<td>virtual void <code>OverwriteScriptManager(const Ptr&lt;Script::ScriptManager&gt; &amp;s)</code></td>
<td>Overwrite the default script manager</td>
</tr>
<tr>
<td>bool <code>IsActive()</code> const</td>
<td>Return true if property is currently active</td>
</tr>
<tr>
<td>virtual void <code>OnLoad()</code></td>
<td>Called from within <code>GameServer::Load()</code> after attributes are loaded</td>
</tr>
<tr>
<td>virtual void <code>OnStart()</code></td>
<td>Called from within <code>GameServer::OnStart()</code> after <code>OnLoad</code> when the complete world exist</td>
</tr>
<tr>
<td>virtual void <code>OnSave()</code></td>
<td>Called from within <code>GameServer::Save()</code> before attributes are saved back to database</td>
</tr>
<tr>
<td>virtual void <code>OnBeginFrame()</code></td>
<td>Called on begin of frame</td>
</tr>
<tr>
<td>virtual void <code>OnFrame()</code></td>
<td>Called in the middle of the feature trigger cycle</td>
</tr>
<tr>
<td>virtual void <code>OnEndFrame()</code></td>
<td>Called at the end of the feature trigger cycle</td>
</tr>
<tr>
<td>virtual void <code>StartRenderDebug()</code></td>
<td>Start render debug</td>
</tr>
<tr>
<td>virtual void <code>OnRenderDebug()</code></td>
<td>Called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void <code>StopRenderDebug()</code></td>
<td>Stop render debug</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void AttachManager (const Ptr&lt; Manager &gt; &amp;manager)</td>
<td>attach a manager to the game world</td>
</tr>
<tr>
<td>virtual void RemoveManager (const Ptr&lt; Manager &gt; &amp;manager)</td>
<td>remove a manager from the game world</td>
</tr>
<tr>
<td>void SetCmdLineArgs (const Util::CommandLineArgs &amp;a)</td>
<td>set command line args</td>
</tr>
<tr>
<td>const Util::CommandLineArgs &amp; GetCmdLineArgs () const</td>
<td>get command line args</td>
</tr>
<tr>
<td>void SetRenderDebug (bool b)</td>
<td>set flag for rendering debug information</td>
</tr>
<tr>
<td>bool HasRenderDebug () const</td>
<td>get flag for rendering debug information</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool Isa (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool Isa (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp;</td>
<td><code>GetClassName</code> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC</code></td>
<td><code>GetClassFourCC</code> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

void Game::FeatureUnit::OnLoad() [virtual, inherited]
called from within GameServer::Load() after attributes are loaded
This method is called from within Game::GameServer::Load() on load of a savegame.

void Game::FeatureUnit::OnStart() [virtual, inherited]
called from within GameServer::OnStart() after OnLoad when the complete world exist
This method is called from within Game::GameServer::OnStart(). Its called after all game features are activated and have initialized their subsystems. Use this

void Game::FeatureUnit::OnSave() [virtual, inherited]
called from within GameServer::Save() before attributes are saved back to database
This method is called from within Game::GameServer::Save(). It's called on save of a game.

void Game::FeatureUnit::OnBeginFrame() [virtual, inherited]
called on begin of frame
This method is called from Game::GameServer::OnBeginFrame() on all game features attached to an GameServer in the order of attachment. Override this method if your FeatureUnit has to do any work at the beginning of the frame.

void
Game::FeatureUnit::OnFrame( ) [virtual, inherited]
called in the middle of the feature trigger cycle

This method is called from Game::GameServer::OnMoveBefore() on all game features attached to an GameServer in the order of attachment. Override this method if your FeatureUnit has any work to do before the physics subsystem is triggered.

void Game::FeatureUnit::OnEndFrame( ) [virtual, inherited]
called at the end of the feature trigger cycle

This method is called from Game::GameServer::OnRender() on all game features attached to an GameServer in the order of attachment. Override this method if your FeatureUnit has any work to do before rendering happens.

void Game::FeatureUnit::OnRenderDebug( ) [virtual, inherited]
called when game debug visualization is on

This method is called from Game::GameServer::OnRenderDebug() on all game features attached to an GameServer in the order of attachment. It's meant for debug issues. It will be called when debug mode is enabled.

void Game::FeatureUnit::AttachManager( const 
  &
  const 
  Manager ) [virtual, inherited]
attach a manager to the game world

Attach a manager object to the game world. The manager's OnActivate() method will be called once right away, and then its OnFrame() method once per frame.

const
void Game::FeatureUnit::RemoveManager(RefPtr<Manager> & manager) [virtual, inherited]

remove a manager from the game world

Remove a manager object from the game world. The manager's OnDeactivate() method will be called.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Shared::CameraSettings
Shared::CameraSettings Class Reference

#include <camerasettings.h>
Detailed Description

Wraps camera settings into an object.

(C) 2009 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CameraSettings ()</strong></td>
<td>default constructor</td>
</tr>
<tr>
<td><strong>void SetupPerspectiveFov (float fov, float aspect, float zNear, float zFar)</strong></td>
<td>setup a perspective view volume</td>
</tr>
<tr>
<td><strong>void SetupOrthogonal (float w, float h, float zNear, float zFar)</strong></td>
<td>setup an orthogonal projection transform</td>
</tr>
<tr>
<td><strong>void UpdateViewMatrix (const Math::matrix44 &amp;m)</strong></td>
<td>update view matrix</td>
</tr>
<tr>
<td><strong>const Math::matrix44 &amp; GetProjTransform () const</strong></td>
<td>get projection matrix</td>
</tr>
<tr>
<td><strong>const Math::matrix44 &amp; GetInvProjTransform () const</strong></td>
<td>get the inverse projection matrix</td>
</tr>
<tr>
<td><strong>const Math::matrix44 &amp; GetViewTransform () const</strong></td>
<td>get view transform (inverse transform)</td>
</tr>
<tr>
<td><strong>const Math::matrix44 &amp; GetViewProjTransform () const</strong></td>
<td>get view projection matrix (non-const!)</td>
</tr>
<tr>
<td><strong>const Math::frustum &amp; GetViewFrustum () const</strong></td>
<td>get view frustum</td>
</tr>
<tr>
<td><strong>const Math::float2 &amp; GetFocalLength () const</strong></td>
<td>get focal length (computed from fov and aspect ratio)</td>
</tr>
<tr>
<td><strong>bool IsPerspective () const</strong></td>
<td>return true if this is a perspective projection</td>
</tr>
<tr>
<td><strong>bool IsOrthogonal () const</strong></td>
<td>return true if this is an orthogonal transform</td>
</tr>
<tr>
<td><strong>float GetZNear () const</strong></td>
<td>get near plane distance</td>
</tr>
<tr>
<td><strong>float GetZFar () const</strong></td>
<td>get far plane distance</td>
</tr>
<tr>
<td><strong>float GetFov () const</strong></td>
<td>get field-of-view (only if perspective)</td>
</tr>
<tr>
<td><strong>float GetAspect () const</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Get Near Width

- **Function**: `GetNearWidth()`
- **Type**: `const float`
- **Description**: Get width of near plane

### Get Near Height

- **Function**: `GetNearHeight()`
- **Type**: `const float`
- **Description**: Get height of near plane

### Get Far Width

- **Function**: `GetFarWidth()`
- **Type**: `const float`
- **Description**: Get width of far plane

### Get Far Height

- **Function**: `GetFarHeight()`
- **Type**: `const float`
- **Description**: Get height of far plane
Member Function Documentation

```c
void
Shared::CameraSettings::SetupPerspectiveFov ( float fov_,
    float aspect_,
    float zNear_,
    float zFar_ )
```

setup a perspective view volume

Setup camera as perspective projection. This method can be called before or after setting up the object. When the object is alive, an update message will be sent to the render-thread.

```c
void
Shared::CameraSettings::SetupOrthogonal ( float w,
    float h,
    float zNear_,
    float zFar_ )
```

setup an orthogonal projection transform

Setup camera as orthogonal projection. This method can be called before or after setting up the object. When the object is alive, an update message will be sent to the render-thread.
Shared::CharJointInfo
Shared::CharJointInfo Class Reference

#include <charjointinfo.h>
Detailed Description

Contains information about a character joint.

(C) 2010 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CharJointInfo()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>CharJointInfo(const Util::StringAtom &amp;jointName, const Math::matrix44 &amp;localMatrix, const Math::matrix44 &amp;globalMatrix)</code></td>
<td>constructor with name and joint index</td>
</tr>
<tr>
<td><code>SetLocalMatrix(const Math::matrix44 &amp;m)</code></td>
<td>set local matrix</td>
</tr>
<tr>
<td><code>GetLocalMatrix()</code></td>
<td>get joint matrix in global space</td>
</tr>
<tr>
<td><code>SetGlobalMatrix(const Math::matrix44 &amp;m)</code></td>
<td>set global matrix</td>
</tr>
<tr>
<td><code>GetGlobalMatrix()</code></td>
<td>get joint matrix in global space</td>
</tr>
<tr>
<td><code>SetJointName(const Util::StringAtom &amp;n)</code></td>
<td>set joint name</td>
</tr>
<tr>
<td><code>GetJointName()</code></td>
<td>get joint name</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Shared:: GraphicsEntityShared
Shared::GraphicsEntityShared Class Reference

#include <graphicsentityshared.h>

Inheritance diagram for Shared::GraphicsEntityShared:
Detailed Description

Shared data object for InternalGraphicsEntity and GraphicsEntity. To be used with a FrameSyncSharedData object.

(C) 2010 Radon Labs GmbH
Public Member Functions

**GraphicsEntityShared ()**

*constructor*
Shared::\textit{ModelEntityShared}
#include <modelentityshared.h>

Inheritance diagram for Shared::ModelEntityShared:

```
Shared::GraphicsEntityShared
  `-> Shared::ModelEntityShared
```
Detailed Description

Shared data object for InternalModelEntity and ModelEntity.

(C) 2010 Radon Labs GmbH
Public Member Functions

**ModelEntityShared ()**

*constructor*
StateObjectFeature::StateGraphicsProperty
StateObjectFeature::StateGraphicsProperty

#include <stategraphicsproperty.h>

Inheritance diagram for StateObjectFeature::StateGraphicsProperty:
Detailed Description

A graphics property which can switch between different visual states.

(C) 2008 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>CallbackType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>callback types</td>
</tr>
</tbody>
</table>
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StateGraphicsProperty ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~StateGraphicsProperty ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual void SetupDefaultAttributes ()</strong></td>
<td>setup default entity attributes</td>
</tr>
<tr>
<td><strong>virtual void OnActivate ()</strong></td>
<td>OnActivate.</td>
</tr>
<tr>
<td><strong>virtual void OnDeactivate ()</strong></td>
<td>called from Entity::DeactivateProperties()</td>
</tr>
<tr>
<td><strong>virtual void SetupAcceptedMessages ()</strong></td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td><strong>virtual void handleMessage (const Ptr<a href="">Messaging::Message</a> &amp;msg)</strong></td>
<td>handle a single message</td>
</tr>
<tr>
<td><strong>virtual void SetupCallbacks ()</strong></td>
<td>setup callbacks for this property, call by entity in OnActivate()</td>
</tr>
<tr>
<td><strong>virtual const Util::String &amp; GetGraphicsResource () const</strong></td>
<td>override to provide a self managed graphics resource (default is Attr::Graphics)</td>
</tr>
<tr>
<td><strong>virtual void OnRenderDebug ()</strong></td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td><strong>const Ptr&lt;Entity&gt; &amp; GetEntity () const</strong></td>
<td>get entity this property is attached to</td>
</tr>
<tr>
<td><strong>bool HasEntity () const</strong></td>
<td>return true if entity pointer is valid</td>
</tr>
<tr>
<td><strong>bool IsActive () const</strong></td>
<td>return true if property is currently active</td>
</tr>
<tr>
<td><strong>virtual void OnLoad ()</strong></td>
<td>called from within Entity::Load() after attributes are loaded</td>
</tr>
<tr>
<td><strong>virtual void OnStart ()</strong></td>
<td>called from within Entity::OnStart() after OnLoad when</td>
</tr>
</tbody>
</table>
virtual void **OnSave** ()
called from within Entity::Save() before attributes are saved back to database

virtual void **OnBeginFrame** ()
called on begin of frame

virtual void **OnMoveBefore** ()
called before movement happens

virtual void **OnMoveAfter** ()
called after movement has happened

virtual void **OnRender** ()
called before rendering happens

virtual void **OnLoseActivity** ()
called when game debug visualization is on

virtual void **OnGainActivity** ()
called when game debug visualization is on

void **AttachHandler** (const **Ptr** < **Handler** > &h)
attach a message handler to the port

void **RemoveHandler** (const **Ptr** < **Handler** > &h)
remove a message handler from the port

void **RemoveAllHandlers** ()
remove all message handler from the port

**SizeT** **GetNumHandlers** () const
return number of handlers attached to the port

const **Ptr** < **Handler** > & **GetHandlerAtIndex** (**IndexT** i) const
get a message handler by index

virtual void **Send** (const **Ptr** < **Message** > &msg)
send a message to the port

const **Util::Array** < const **Id** * > & **GetAcceptedMessages** () const
get the array of accepted messages (sorted)

**bool** **AcceptsMessage** (const **Id** &msgId) const
return true if port accepts this msg

**int** **GetRefCount** () const
get the current refcount

void **AddRef** ()
increment refcount by one
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void OnSwitchActiveState(const Util::String &amp;stateName)</code></td>
<td>called when a SwitchActiveState message is received</td>
</tr>
<tr>
<td><code>void UpdateTransforms()</code></td>
<td>update graphics entities transforms</td>
</tr>
<tr>
<td><code>void OnLoadResources(const Util::Dictionary&lt;Util::String, StateInfo&gt; &amp;resources)</code></td>
<td>load all entities</td>
</tr>
<tr>
<td><code>virtual void SetupGraphicsEntities()</code></td>
<td>setup graphics entities</td>
</tr>
<tr>
<td><code>virtual void UpdateTransform(const Math::matrix44 &amp;m, bool setDirectly=false)</code></td>
<td>update the graphics entity's transform</td>
</tr>
<tr>
<td><code>void SetVisible(bool visible)</code></td>
<td>set visible</td>
</tr>
<tr>
<td><code>void OnSetOverwriteColor(const Ptr&lt;SetOverwriteColor&gt; &amp;msg)</code></td>
<td>on set overwrite color</td>
</tr>
<tr>
<td><code>void OnSetShaderVariable(const Ptr&lt;SetShaderVariable&gt; &amp;msg)</code></td>
<td>set shader variable</td>
</tr>
<tr>
<td><code>void SetEntity(const Ptr&lt;Entity&gt; &amp;v)</code></td>
<td>Set entity, this is attached to, to <code>v</code>.</td>
</tr>
<tr>
<td><code>void ClearEntity()</code></td>
<td>Remove entity.</td>
</tr>
<tr>
<td><code>void RegisterMessage(const Id &amp;msgId)</code></td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void StateObjectFeature::StateGraphicsProperty::OnActivate() [virtual]
```

OnActivate.

Attach the property to a game entity. This will create and setup the required graphics entities.

Reimplemented from `GraphicsFeature::GraphicsProperty`.

```cpp
void StateObjectFeature::StateGraphicsProperty::UpdateTransforms() [protected]
```

update graphics entities transforms

This transfers the physics entity transforms into the graphics entity transforms.

```cpp
const Util::String &
GraphicsFeature::GraphicsProperty::GetGraphicsResource() const [inline, virtual, inherited]
```

override to provide a self managed graphics resource (default is `Attr::Graphics`)

Get the default graphics resource, which is `Attr::Graphics`. subclasses may override this to provide a self managed resource.

```cpp
void GraphicsFeature::GraphicsProperty::SetupGraphicsEntities() [protected, virtual, inherited]
```

setup graphics entities

Setup the graphics entities. You may override this method in a subclass if different setup is needed.

Reimplemented in `GraphicsFeature::ActorGraphicsProperty`.

```cpp
void const
```
GraphicsFeature::GraphicsProperty::UpdateTransform (Math::matrix44 \( m \),
&
bool \( \text{setDirectly} \) = false

[protected, virtual, inherited]

update the graphics entity's transform

Called to update the graphics entity's transform.

void
GraphicsFeature::GraphicsProperty::SetVisible (bool \( \text{visible} \ )) [protected, inherited]

Shows or hides all attached graphics entities.

void
Game::Property::OnLoad ( ) [virtual, inherited]

called from within Entity::Load() after attributes are loaded

This method is called from within Game::Entity::Load() after the entity attributes have been loaded from the database. You can override this method in a subclass if further initialization is needed for the property after attributes have been loaded from the database, but please be aware that this method may not be called if the entity is created directly.

Reimplemented in PhysicsFeature::TriggerProperty, and
StateObjectFeature::StateProperty.

void
Game::Property::OnStart ( ) [virtual, inherited]

called from within Entity::OnStart() after OnLoad when the complete world exist

This method is called from within Game::Entity::OnStart(). This is the moment when the world is complete and the entity can establish connections to other entities.

Reimplemented in GraphicsFeature::CameraProperty.
void Game::Property::OnSave() [virtual, inherited]  

called from within Entity::Save() before attributes are saved back to database

This method is called from within Game::Entity::Save() before the entity attributes will be saved to the database. You can override this method in a subclass if actions are needed before a save happens (this is usually the case if entity attributes need to be updated by the property before saving).

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void Game::Property::OnBeginFrame() [virtual, inherited]  

called on begin of frame

This method is called from Game::Entity::OnBeginFrame() on all properties attached to an entity in the order of attachment. Override this method if your property has to do any work at the beginning of the frame.

Reimplemented in PhysicsFeature::TriggerProperty, and StateObjectFeature::StateProperty.

void Game::Property::OnMoveBefore() [virtual, inherited]  

called before movement happens

This method is called from Game::Entity::OnMoveBefore() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before the physics subsystem is triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty.
called after movement has happened

This method is called from `Game::Entity::OnMoveAfter()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`, and `PhysicsFeature::PhysicsProperty`.

```cpp
void Game::Property::OnMoveAfter()
```

called before rendering happens

This method is called from `Game::Entity::OnRender()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before rendering happens.

Reimplemented in `GraphicsFeature::CameraProperty`, `GraphicsFeature::ChaseCameraProperty`, and `GraphicsFeature::MayaCameraProperty`.

```cpp
void Game::Property::OnRender()
```

called when game debug visualization is on

This method is called from `Game::Entity::OnLoseActivity()` on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`.

```cpp
void Game::Property::OnLoseActivity()
```

called after rendering has happened

This method is called from `Game::Entity::OnGainActivity()` on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after rendering happens.

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`.

```cpp
void Game::Property::OnGainActivity()
```
called when game debug visualization is on

This method is called from `Game::Entity::OnRenderDebug()` on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

```cpp
void Messaging::Port::AttachHandler(const Ptr<Handler> & h) [inherited]
```

attach a message handler to the port

```cpp
void Messaging::Port::RemoveHandler(const Ptr<Handler> & h) [inherited]
```

remove a message handler from the port

```cpp
void Messaging::Port::Send(const Ptr<Message> & msg) [virtual, inherited]
```

send a message to the port

Send a message to the port. This will immediately call the `HandleMessage()` method of all attached handlers. If the message has been handled by at least one of the handlers, the `Handled()` flag of the message will be set to true.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount
Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
StateObjectFeature::StateInfo
StateObjectFeature::StateInfo Class Reference

#include <stateinfo.h>
### Data Fields

<table>
<thead>
<tr>
<th>Util::String</th>
<th>soundName</th>
</tr>
</thead>
<tbody>
<tr>
<td>constructor</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
StateObjectFeature::StateObjectFeatureUnit
StateObjectFeature::StateObjectFeatureU
Class Reference

#include <stateobjectfeatureunit.h>
Detailed Description

(C) 2008 Radon Labs GmbH

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**StateObjectFeature::StateProperty**
StateObjectFeature::StateProperty
Class Reference

#include <stateproperty.h>

Inheritance diagram for StateObjectFeature::StateProperty:
Detailed Description

Central property which manages "state switches". These are just visual/physical state switches, not behavioral state switches. Usually works together with a StatePhysicsProperty and StateGraphicsProperty.

FIXME: need to implement advanced stuff: Play Sound, Transition States.

(C) 2008 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>CallbackType</th>
<th>callback types</th>
</tr>
</thead>
</table>

## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StateProperty ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual void SetupCallbacks ()</strong></td>
<td>setup callbacks for this property</td>
</tr>
<tr>
<td><strong>virtual void SetupDefaultAttributes ()</strong></td>
<td>setup default entity attributes</td>
</tr>
<tr>
<td><strong>virtual void OnLoad ()</strong></td>
<td>called from within Entity::Load() after attributes are loaded</td>
</tr>
<tr>
<td><strong>virtual void OnSave ()</strong></td>
<td>called from within Entity::Save() before attributes are saved back to database</td>
</tr>
<tr>
<td><strong>virtual void SetupAcceptedMessages ()</strong></td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td><strong>virtual void HandleMessage (const Ptr<a href="">Messaging::Message</a> &amp;msg)</strong></td>
<td>handle a single message</td>
</tr>
<tr>
<td><strong>virtual void OnBeginFrame ()</strong></td>
<td>called on begin of frame</td>
</tr>
<tr>
<td><strong>virtual void OnDeactivate ()</strong></td>
<td>called from Entity::DeactivateProperties()</td>
</tr>
<tr>
<td><strong>void SetFileName (const Util::String &amp;name)</strong></td>
<td>sets the filename of the ui.xml file</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetFileName () const</strong></td>
<td>gets the filename</td>
</tr>
<tr>
<td><strong>void ParseFile ()</strong></td>
<td>opens the file and creates the gui structure</td>
</tr>
<tr>
<td><strong>const Ptr&lt;Entity&gt; &amp; GetEntity () const</strong></td>
<td>get entity this property is attached to</td>
</tr>
<tr>
<td><strong>bool HasEntity () const</strong></td>
<td>return true if entity pointer is valid</td>
</tr>
<tr>
<td><strong>virtual void OnActivate ()</strong></td>
<td>called from Entity::ActivateProperties()</td>
</tr>
<tr>
<td><strong>bool IsActive () const</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void OnStart ()</td>
<td>called from within Entity::OnStart() after OnLoad when the complete world exist</td>
</tr>
<tr>
<td>virtual void OnMoveBefore ()</td>
<td>called before movement happens</td>
</tr>
<tr>
<td>virtual void OnMoveAfter ()</td>
<td>called after movement has happened</td>
</tr>
<tr>
<td>virtual void OnRender ()</td>
<td>called before rendering happens</td>
</tr>
<tr>
<td>virtual void OnRenderDebug ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void OnLoseActivity ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void OnGainActivity ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>void AttachHandler (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td>void RemoveHandler (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td>void RemoveAllHandlers ()</td>
<td>remove all message handler from the port</td>
</tr>
<tr>
<td>SizeT GetNumHandlers () const</td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td>const Ptr&lt; Handler &gt; &amp; GetHandlerAtIndex (IndexT i) const</td>
<td>get a message handler by index</td>
</tr>
<tr>
<td>virtual void Send (const Ptr&lt;Message&gt; &amp;msg)</td>
<td>send a message to the port</td>
</tr>
<tr>
<td>const Util::Array&lt;const Id *&gt; &amp; GetAcceptedMessages () const</td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td>bool AcceptsMessage (const Id &amp;msgId)</td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA(const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName()</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC()</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
</table>

`dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)`
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetEntity (const Ptr&lt; Entity &gt; &amp;v)</code></td>
<td>Set entity, this is attached to, to <code>v</code>.</td>
</tr>
<tr>
<td><code>void ClearEntity ()</code></td>
<td>Remove entity.</td>
</tr>
<tr>
<td><code>void RegisterMessage (const Id &amp;msgId)</code></td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Member Function Documentation

**void**
_stateobjectfeature::stateproperty::onbeginframe() [virtual]

called on begin of frame

This checks whether we are in a transition state and whether we need to switch to the next state.

Reimplemented from **Game::Property**.

**void**
_game::property::onactivate() [virtual, inherited]

called from **entity::activateproperties()**

This method is called by **Game::Entity::ActivateProperties()**. Use this method for one-time initializations of the property.


**void**
_game::property::onstart() [virtual, inherited]

called from within **entity::onstart()** after **onload** when the complete world exist

This method is called from within **Game::Entity::OnStart()**. This is the moment when the world is complete and the entity can establish connections to other entities.

Reimplemented in **GraphicsFeature::CameraProperty**.
void
Game::Property::OnMoveBefore() [virtual, inherited]
called before movement happens

This method is called from Game::Entity::OnMoveBefore() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before the physics subsystem is triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty.

void
Game::Property::OnMoveAfter() [virtual, inherited]
called after movement has happened

This method is called from Game::Entity::OnMoveAfter() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do after the physics subsystem has been triggered.

Reimplemented in PhysicsFeature::ActorPhysicsProperty, and PhysicsFeature::PhysicsProperty.

void
Game::Property::OnRender() [virtual, inherited]
called before rendering happens

This method is called from Game::Entity::OnRender() on all properties attached to an entity in the order of attachment. Override this method if your property has any work to do before rendering happens.

Reimplemented in GraphicsFeature::CameraProperty, GraphicsFeature::ChaseCameraProperty, and GraphicsFeature::MayaCameraProperty.

void
Game::Property::OnRenderDebug() [virtual, inherited]
called when game debug visualization is on

This method is called from `Game::Entity::OnRenderDebug()` on all properties attached to an entity in the order of attachment. It's meant for debug issues. It will be called when debug mode is enabled.

Reimplemented in `GraphicsFeature::GraphicsProperty`, `PhysicsFeature::ActorPhysicsProperty`, and `PhysicsFeature::TriggerProperty`.

```cpp
void Game::Property::OnLoseActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from `Game::Entity::OnLoseActivity()` on all properties attached to an entity in the order of attachment. It indicates that the entity will no longer be trigger, due to leaving the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

Reimplemented in `PhysicsFeature::ActorPhysicsProperty`.

```cpp
void Game::Property::OnGainActivity() [virtual, inherited]
```
called when game debug visualization is on

This method is called from `Game::Entity::OnRenderDebug()` on all properties attached to an entity in the order of attachment. It indicates that the entity will be trigger from now on, due to entering the "Activity Bubble", i.e. the area of interest (most probably around the active camera).

```cpp
void Messaging::Port::AttachHandler(const std::shared_ptr<Handler> & h ) [inherited]
```
attach a message handler to the port

Attach a message handler to the port.
void Messaging::Port::RemoveHandler ( const Ptr< Handler >& h ) [inherited]

remove a message handler from the port

Remove a message handler from the port.

void Messaging::Port::Send ( const Ptr< Message >& msg ) [virtual, inherited]

send a message to the port

Send a message to the port. This will immediately call the HandleMessage() method of all attached handlers. If the message has been handled by at least one of the handlers, the Handled() flag of the message will be set to true.

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String & Core::RefCounted::GetClassName

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
System::AppEntry
System::AppEntry Class Reference

#include <appentry.h>
Detailed Description

Implements a platform-independent app-entry point. In your main file, put the line

ImplementNebulaApplication();

And then replace your main() function with:

void NebulaMain(const CommandLineArgs& args)

(C) 2007 Radon Labs GmbH
System::ByteOrder
System::ByteOrder Class Reference

#include <byteorder.h>
Detailed Description

Provides information and methods to help with byte ordering issues.

The following byte orders are defined:

- ByteOrder::LittleEndian: Intel byte order
- ByteOrder::BigEndian: Motorola byte order
- ByteOrder::Network network byte order (always big endian)
- ByteOrder::Host hardwired to little or big endian depending on host cpu

NOTE: on console-platforms it is strongly recommended to store binary data in console-byte-order during asset processing on the development PC, so that the console doesn't waste loading time with byte conversions!

(C) 2007 Radon Labs GmbH
Public Member Functions

**ByteOrder ()**
*default constructor*

**ByteOrder (Type fromByteOrder, Type toByteOrder)**
*constructor: set byte order conversion rule*

void **SetFromByteOrder** (Type fromByteOrder)
*set from-byte-order*

Type **GetFromByteOrder () const**
*get from-byte-order*

void **SetToByteOrder** (Type toByteOrder)
*set to-byte-order*

Type **GetToByteOrder () const**
*get to-byte-order*

template<class TYPE>
void **ConvertInPlace** (TYPE &val) const
*endian-convert in place*

template<class TYPE>
TYPE **Convert** (TYPE val) const
*endian-convert by copy*
Static Public Member Functions

template<class TYPE>
    static void ConvertInPlace (Type fromByteOrder, Type toByteOrder, TYPE &val)
        endian-convert in place

template<class TYPE>
static TYPE Convert (Type fromByteOrder, Type toByteOrder, TYPE val)
        endian-convert by copy
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

System::Cpu
System::Cpu Class Reference

#include <cpu.h>
Detailed Description

Provides information about the system's CPU(s).

(C) 2007 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**System::SystemInfo**
System::SystemInfo Class Reference

#include <systeminfo.h>

Inheritance diagram for System::SystemInfo:
Detailed Description

Provides information about the host system.

(C) 2008 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**System::Win32Environment**
System::Win32Environment Class Reference

#include <win32environment.h>
Detailed Description

Provides read-access to environment variables. Useful for tools. NOTE: using this class restricts your code to the Win32 platform.

(C) 2009 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Exists</code></td>
<td><code>static bool Exists(const Util::String &amp;envVarName)</code></td>
<td>return true if env-variable exists</td>
</tr>
<tr>
<td><code>Read</code></td>
<td><code>static Util::String Read(const Util::String &amp;envVarName)</code></td>
<td>get value of existing env-variable</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

System::Win32Registry
System::Win32Registry Class Reference

#include <win32registry.h>
Detailed Description

A simple wrapper class to access the Win32 registry. NOTE: using this class restricts your code to the Win32 platform.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>RootKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>enumeration</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static bool Exists (RootKey rootKey, const Util::String &amp;key, const Util::String &amp;name)</code></td>
<td>return true if a registry entry exists</td>
</tr>
<tr>
<td><code>static bool WriteString (RootKey rootKey, const Util::String &amp;key, const Util::String &amp;name, const Util::String &amp;value)</code></td>
<td>write a registry entry</td>
</tr>
<tr>
<td><code>static Util::String ReadString (RootKey rootKey, const Util::String &amp;key, const Util::String &amp;name)</code></td>
<td>read a string registry entry, the string will be UTF-8 encoded!</td>
</tr>
<tr>
<td><code>static int ReadInt (RootKey rootKey, const Util::String &amp;key, const Util::String &amp;name)</code></td>
<td>read an int registry entry</td>
</tr>
<tr>
<td><code>static bool Delete (RootKey rootKey, const Util::String &amp;key)</code></td>
<td>delete a registry key (and all its contained values)</td>
</tr>
<tr>
<td><code>static RootKey AsRootKey (const Util::String &amp;str)</code></td>
<td>convert rootkey from string</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool System::Win32Registry::Exists(
    const RootKey rootKey,
    const Util::String key,
    const Util::String name
) [static]
```

Return true if a registry entry exists

Return true if a specific entry exists in the registry. To check only for the existence of a key without the contained value, pass an empty 'name' string.

```cpp
bool System::Win32Registry::WriteString(
    const RootKey rootKey,
    const Util::String key,
    const Util::String name,
    const Util::String value
) [static]
```

Write a registry entry

Set a key value in the registry. This will create the key if it doesn't exist.

```cpp
String System::Win32Registry::ReadString(
    const RootKey rootKey,
    const Util::String key,
    const Util::String name
)
```
read a string registry entry, the string will be UTF-8 encoded!

Get a string value from the registry. Fails hard if the key doesn't exists (use the `Exists()` method to make sure that the key exists!). NOTE that this method returns an UTF-8 encoded string!

```cpp
int System::Win32Registry::ReadInt ( RootKey rootKey,
                             const Util::String key,
                             &
                             const Util::String name
                             &
) )  [static]
```

read an int registry entry

Get an int value from the registry. Fails hard if the key doesn't exists (use the `Exists()` method to make sure that the key exists!).

```cpp
bool System::Win32Registry::Delete ( RootKey rootKey,
                             const Util::String key
                             &
) )  [static]
```

delete a registry key (and all its contained values)

This deletes a complete registry key with all its values.

```cpp
Win32Registry::RootKey System::Win32Registry::AsRootKey ( Util::String str ) [static]
```

convert rootkey from string

Converts a string (all capitels, e.g. HKEY_CURRENT_USER) into a RootKey value.
Threading::Barrier
Threading::Barrier Class Reference

#include <barrier.h>
Detailed Description

Implements the 2 macros ReadWriteBarrier and MemoryBarrier.

ReadWriteBarrier prevents the compiler from re-ordering memory accesses across the barrier.

MemoryBarrier prevents the CPU from reordering memory access across the barrier (all memory access will be finished before the barrier is crossed).

(C) 2007 Radon Labs GmbH
Threading::CriticalSection
#include <criticalsection.h>

Inheritance diagram for Threading::CriticalSection:
Detailed Description

Critical section objects are used to protect a portion of code from parallel execution. Define a static critical section object and use its Enter() and Leave() methods to protect critical sections of your code.

(C) 2006 Radon Labs GmbH
Threading::Event
Threading::Event Class Reference

#include <event.h>

Inheritance diagram for Threading::Event:

```
Win32::Win32Event
  ↑
Threading::Event
```
Detailed Description

Todo:
   describe Event class

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Signal ()</code></td>
<td>signal the event</td>
</tr>
<tr>
<td><code>Reset ()</code></td>
<td>reset the event (only if manual reset)</td>
</tr>
<tr>
<td><code>Wait () const</code></td>
<td>wait for the event to become signalled</td>
</tr>
<tr>
<td><code>WaitTimeout (int ms) const</code></td>
<td>wait for the event with timeout in millisecs</td>
</tr>
<tr>
<td><code>Peek () const</code></td>
<td>check if event is signalled</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool Win360::Win360Event::WaitTimeout ( int timeoutInMilliSec ) const [inline, inherited]

wait for the event with timeout in millisecs

Waits for the event to become signaled with a specified timeout in milliseconds. If the method times out it will return false, if the event becomes signalled within the timeout it will return true.

bool Win360::Win360Event::Peek ( ) const [inline, inherited]

check if event is signalled

This checks if the event is signalled and returns immediately.
Threading::Interlocked
Threading::Interlocked Class Reference

#include <interlocked.h>

Inheritance diagram for Threading::Interlocked:
Detailed Description

Provide simple atomic operations on memory variables.

(C) 2006 Radon Labs GmbH
Threading::ObjectRef
Threading::ObjectRef Class Reference

#include <objectref.h>

Inheritance diagram for Threading::ObjectRef:
Detailed Description

A thread-safe reference to a shared object. Object refs are used with the messaging system to reference opaque objects created and manipulated in other threads. **ObjectRef** objects must be created on the heap (thus they are ref-counted) because the "client-side" owner-object may be discarded before the target object in the other thread is destroyed.

(C) 2010 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ObjectRef ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~ObjectRef ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>bool IsValid ()</code> const</td>
<td>return true if the <code>ObjectRef</code> points to a valid object</td>
</tr>
<tr>
<td><code>template&lt;class TYPE&gt; void Validate (TYPE *obj)</code></td>
<td>validate the ref with a pointer to a target object (must be RefCounted)</td>
</tr>
<tr>
<td><code>void Invalidate ()</code></td>
<td>invalidate the ref</td>
</tr>
<tr>
<td><code>template&lt;class TYPE&gt; TYPE * Ref ()</code> const</td>
<td>get pointer to object</td>
</tr>
<tr>
<td><code>int GetRefCount ()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code> const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</code> const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName)</code> const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC)</code> const</td>
<td></td>
</tr>
</tbody>
</table>
return true if this object is instance of given class, or a derived class, by fourcc

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount
Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one
Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
get the class name
Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
This method should be called as the very last before an application exits.
Threading::SafeFlag
Threading::SafeFlag Class Reference

#include <safeflag.h>
Detailed Description

A thread-safe flag variable.

(C) 2008 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SafeFlag()</td>
<td>constructor</td>
</tr>
<tr>
<td>void Set()</td>
<td>set the flag</td>
</tr>
<tr>
<td>void Clear()</td>
<td>clear the flag</td>
</tr>
<tr>
<td>bool Test() const</td>
<td>test if the flag is set</td>
</tr>
<tr>
<td>bool TestAndClearIfSet()</td>
<td>test if flag is set, if yes, clear flag</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:50 2010
Threading::SafePriorityQueue
#include <safepriorityqueue.h>

Inheritance diagram for Threading::SafePriorityQueue< PRITYPE, TYPE >:

```
Util::Queue< Util::KeyValuePair< PRITYPE, TYPE > >
```

```
< PRITYPE, TYPE >._map
```

```
< PRITYPE, TYPE >._map
```

```
< Util::KeyValuePair< PRITYPE, TYPE > >
```

```
shape=rect coords="0,0,314,24"
```
Detailed Description

template<class PRITYPE, class TYPE>
class Threading::SafePriorityQueue< PRITYPE, TYPE >

A thread-safe priority-sorted queue which protects itself with critical sections. Offers a method to wait for new elements to be added. Useful for inter-thread communications.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SafePriorityQueue ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>SafePriorityQueue (const SafePriorityQueue&lt; PRITYPE, TYPE &gt; &amp;rhs)</strong></td>
<td>Copy constructor</td>
</tr>
<tr>
<td><strong>operator= (const SafePriorityQueue&lt; PRITYPE, TYPE &gt; &amp;rhs)</strong></td>
<td>Assignment operator</td>
</tr>
<tr>
<td><strong>SizeT Size () const</strong></td>
<td>returns number of elements in the queue</td>
</tr>
<tr>
<td><strong>bool IsEmpty () const</strong></td>
<td>return true if queue is empty</td>
</tr>
<tr>
<td><strong>void Clear ()</strong></td>
<td>remove all elements from the queue</td>
</tr>
<tr>
<td><strong>void Insert (PRITYPE pri, const TYPE &amp;e)</strong></td>
<td>add element to the back of the queue</td>
</tr>
<tr>
<td><strong>void EraseMatchingElements (const TYPE &amp;e)</strong></td>
<td>erase all matching elements</td>
</tr>
<tr>
<td><strong>TYPE Dequeue ()</strong></td>
<td>remove the element from the front of the queue</td>
</tr>
<tr>
<td><strong>TYPE Peek () const</strong></td>
<td>get copy of element at front of queue without removing it</td>
</tr>
<tr>
<td><strong>void Wait ()</strong></td>
<td>wait until queue contains at least one element</td>
</tr>
<tr>
<td><strong>void Signal ()</strong></td>
<td>signal the internal event, so that <code>Wait()</code> will return</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>operator[] (IndexT index) const</strong></td>
<td>access element by index, 0 is the frontmost element (next to be dequeued)</td>
</tr>
<tr>
<td><strong>bool operator==(const Queue&lt;Util::KeyValuePair&lt;PRITYPE, TYPE&gt; &amp;&gt; &amp;rhs) const</strong></td>
<td>equality operator</td>
</tr>
<tr>
<td><strong>bool operator!=(const Queue&lt;Util::KeyValuePair&lt;PRITYPE, TYPE&gt; &amp;&gt; &amp;rhs) const</strong></td>
<td>inequality operator</td>
</tr>
<tr>
<td><strong>void Reserve(SizeT num)</strong></td>
<td>increase capacity to fit N more elements into the queue</td>
</tr>
<tr>
<td><strong>bool Contains(const Util::KeyValuePair&lt;PRITYPE, TYPE&gt; &amp;e) const</strong></td>
<td>return true if queue contains element</td>
</tr>
<tr>
<td><strong>void Enqueue(const Util::KeyValuePair&lt;PRITYPE, TYPE&gt; &amp;e)</strong></td>
<td>add element to the back of the queue</td>
</tr>
</tbody>
</table>
Member Function Documentation

template<class PRITYPE, class TYPE>
void 
Threading::SafePriorityQueue< ()
PRITYPE, TYPE >::Signal

signal the internal event, so that Wait() will return

This signals the internal event object, on which Wait() may be waiting. This method may be useful to wake up a thread waiting for events when it should stop.
#include <safequeue.h>

Inheritance diagram for Threading::SafeQueue< TYPE >:

```
Util::Queue< TYPE >

Threading::SafeQueue< TYPE >
```
Detailed Description

template<class TYPE>
class Threading::SafeQueue< TYPE >

Thread-safe version of Util::Queue. The SafeQueue is normally configured to signal an internal Event object when an element is enqueued, so that a worker-thread can wait for new elements to arrive. This is the default behaviour. This doesn't make sense for a continuously running thread (i.e. a rendering thread), thus this behaviour can be disabled using the SetSignalOnEnqueueEnabled(). In this case, the Enqueue() method won't signal the internal event, and the Wait() method will return immediately without ever waiting.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SafeQueue ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>SafeQueue (const SafeQueue&lt; TYPE &gt; &amp;rhs)</strong></td>
<td>copy constructor</td>
</tr>
<tr>
<td><strong>void operator= (const SafeQueue&lt; TYPE &gt; &amp;rhs)</strong></td>
<td>assignment operator</td>
</tr>
<tr>
<td><strong>void SetSignalOnEnqueueEnabled (bool b)</strong></td>
<td>enable/disable signalling on <code>Enqueue()</code> (default is enabled)</td>
</tr>
<tr>
<td><strong>bool IsSignalOnEnqueueEnabled () const</strong></td>
<td>return signalling-on-Enqueue() flag</td>
</tr>
<tr>
<td><strong>SizeT Size () const</strong></td>
<td>returns number of elements in the queue</td>
</tr>
<tr>
<td><strong>bool IsEmpty () const</strong></td>
<td>return true if queue is empty</td>
</tr>
<tr>
<td><strong>void Clear ()</strong></td>
<td>remove all elements from the queue</td>
</tr>
<tr>
<td><strong>void Enqueue (const TYPE &amp;e)</strong></td>
<td>add element to the back of the queue</td>
</tr>
<tr>
<td><strong>void EnqueueArray (const Util::Array&lt; TYPE &gt; &amp;a)</strong></td>
<td>enqueue an array of elements</td>
</tr>
<tr>
<td><strong>TYPE Dequeue ()</strong></td>
<td>remove the element from the front of the queue</td>
</tr>
<tr>
<td><strong>void DequeueAll (Util::Array&lt; TYPE &gt; &amp;outArray)</strong></td>
<td>dequeue all events (only requires one lock)</td>
</tr>
<tr>
<td><strong>TYPE Peek () const</strong></td>
<td>access to element at front of queue without removing it</td>
</tr>
<tr>
<td><strong>void Wait ()</strong></td>
<td>wait until queue contains at least one element</td>
</tr>
<tr>
<td><strong>void WaitTimeout (int ms)</strong></td>
<td>wait until queue contains at least one element, or time-out happens</td>
</tr>
<tr>
<td><strong>void Signal ()</strong></td>
<td>signal the internal event, so that <code>Wait()</code> will return</td>
</tr>
<tr>
<td><strong>void EraseMatchingElements (const TYPE &amp;e)</strong></td>
<td></td>
</tr>
</tbody>
</table>
erase all matching elements
## Protected Member Functions

<table>
<thead>
<tr>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE &amp; \textbf{operator[]} (IndexT index) const</td>
<td>access element by index, 0 is the frontmost element (next to be dequeued)</td>
</tr>
<tr>
<td>bool \textbf{operator==} (const Queue&lt; TYPE &gt; &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool \textbf{operator!=} (const Queue&lt; TYPE &gt; &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td>void \textbf{Reserve} (SizeT num)</td>
<td>increase capacity to fit N more elements into the queue</td>
</tr>
<tr>
<td>bool \textbf{Contains} (const TYPE &amp;e) const</td>
<td>return true if queue contains element</td>
</tr>
</tbody>
</table>
Member Function Documentation

template<class TYPE>
void
Thread::SafeQueue< ( )
TYPE >::Signal

signal the internal event, so that **Wait()** will return

This signals the internal event object, on which **Wait()** may be waiting. This method may be useful to wake up a thread waiting for events when it should stop.
Threading::Thread
#include <thread.h>

Inheritance diagram for Threading::Thread:

```
Core::RefCounted
  ↓
Win32::Win32Thread
  ↓
Threading::Thread
  ↓
Jobs::TPWorkerThread  Messaging::HandlerThreadBase
  ↓  ↓
FrameSync::FrameSyncHandlerThread  Messaging::BlockingHandlerThread  Messaging::RunThroughHandlerThread
```
Detailed Description

Todo:
   describe Thread class

(C) 2006 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>thread priorities</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetPriority (Priority p)</code></td>
<td>Set the thread priority</td>
</tr>
<tr>
<td><code>Priority GetPriority () const</code></td>
<td>Get the thread priority</td>
</tr>
<tr>
<td><code>void SetCoreId (System::Cpu::CoreId coreId)</code></td>
<td>Set CPU core on which the thread should be running</td>
</tr>
<tr>
<td><code>System::Cpu::CoreId GetCoreId () const</code></td>
<td>Get the CPU core on which the thread should be running</td>
</tr>
<tr>
<td><code>void SetStackSize (SizeT s)</code></td>
<td>Set stack size in bytes (default is 4 KByte)</td>
</tr>
<tr>
<td><code>SizeT GetStackSize () const</code></td>
<td>Get stack size</td>
</tr>
<tr>
<td><code>void SetName (const Util::String &amp;n)</code></td>
<td>Set thread name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetName () const</code></td>
<td>Get thread name</td>
</tr>
<tr>
<td><code>void Start ()</code></td>
<td>Start executing the thread code, returns when thread has actually started</td>
</tr>
<tr>
<td><code>void Stop ()</code></td>
<td>Request threading code to stop, returns when thread has actually finished</td>
</tr>
<tr>
<td><code>bool IsRunning () const</code></td>
<td>Return true if thread has been started</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>Function Description</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Static void YieldThread ()</strong></td>
<td>yield the thread (gives up current time slice)</td>
</tr>
<tr>
<td>*<em>static void SetMyThreadName (const char <em>n)</em></em></td>
<td>set thread name from within thread context</td>
</tr>
<tr>
<td><strong>static const char * GetMyThreadName ()</strong></td>
<td>obtain name of thread from within thread context</td>
</tr>
<tr>
<td><strong>static Threading::ThreadId GetMyThreadId ()</strong></td>
<td>get the thread ID of this thread</td>
</tr>
<tr>
<td><strong>static void DumpRefCountingLeaks ()</strong></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual void EmitWakeupSignal()</code></td>
<td><em>override this method if your thread loop needs a wakeup call before stopping</em></td>
</tr>
<tr>
<td><code>virtual void DoWork()</code></td>
<td><em>this method runs in the thread context</em></td>
</tr>
<tr>
<td><code>bool ThreadStopRequested()</code></td>
<td><em>check if stop is requested, call from DoWork() to see if the thread proc should quit</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Win360::Win360Thread::SetName ( const Util::String & n ) [inline, inherited]
```

set thread name

Set the thread's name. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

```cpp
const Util::String & Win360::Win360Thread::GetName ( ) const [inline, inherited]
```

get thread name

Get the thread's name. This is the vanilla method which returns the name member. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

```cpp
void Win360::Win360Thread::Start ( ) [inherited]
```

start executing the thread code, returns when thread has actually started

Start the thread, this creates a Win32 thread and calls the static `ThreadProc`, which in turn calls the virtual `DoWork()` class of this object. The method waits for the thread to start and then returns.

```cpp
void Win360::Win360Thread::Stop ( ) [inherited]
```

request threading code to stop, returns when thread has actually finished

This stops the thread by signalling the stopRequestEvent and waits for the thread to actually quit. If the thread code runs in a loop it should
use the IsStopRequested() method to see if the thread object wants it to shutdown. If so **DoWork()** should simply return.

Reimplemented in **Jobs::TPWorkerThread**.

```cpp
bool Win360::Win360Thread::IsRunning() const [inherited]
```

return true if thread has been started

Returns true if the thread is currently running.

```cpp
void Win360::Win360Thread::YieldThread() [static, inherited]
```

yield the thread (gives up current time slice)

The yield function is empty on **Win32** and Xbox360.

```cpp
void Win360::Win360Thread::SetMyThreadName(const char * n) [static, inherited]
```

set thread name from within thread context

Static method which sets the name of this thread. This is called from within ThreadProc. The string pointed to must remain valid until the thread is terminated!

```cpp
const char * Win360::Win360Thread::GetMyThreadName() [static, inherited]
```

obtain name of thread from within thread context

Static method to obtain the current thread name from anywhere in the thread's code.

```cpp
Threading::ThreadId Win360::Win360Thread::GetMyThreadId() [static, inherited]
```

get the thread ID of this thread
Static method which returns the ThreadId of this thread.

```cpp
void Win360::Win360Thread::EmitWakeupSignal() [protected, virtual, inherited]
```

override this method if your thread loop needs a wakeup call before stopping

This method is called by `Thread::Stop()` after setting the stopRequest event and before waiting for the thread to stop. If your thread runs a loop and waits for jobs it may need an extra wakeup signal to stop waiting and check for the `ThreadStopRequested()` event. In this case, override this method and signal your event object.

Reimplemented in `Jobs::TPWorkerThread`, and `Messaging::BlockingHandlerThread`.

```cpp
void Win360::Win360Thread::DoWork() [protected, virtual, inherited]
```

this method runs in the thread context

This method should be derived in a Thread subclass and contains the actual code which is run in the thread. The method must not call C-Lib functions under `Win32`. To terminate the thread, just return from this function. If `DoWork()` runs in an infinite loop, call `ThreadStopRequested()` to check whether the Thread object wants the thread code to quit.

Reimplemented in `FrameSync::FrameSyncHandlerThread`, `Jobs::TPWorkerThread`, `Messaging::BlockingHandlerThread`, and `Messaging::RunThroughHandlerThread`.

```cpp
bool Win360::Win360Thread::ThreadStopRequested() const [inline, protected, inherited]
```

check if stop is requested, call from `DoWork()` to see if the thread proc should quit

If the derived `DoWork()` method is running in a loop it must regularly check if the process wants the thread to terminate by calling
ThreadStopRequested() and simply return if the result is true. This will cause the thread to shut down.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Threading::ThreadBarrier
#include <threadbarrier.h>

Inheritance diagram for Threading::ThreadBarrier:

```
Win32::Win32ThreadBarrier
    |
    V
Threading::ThreadBarrier
```
Detailed Description

Block until all thread have arrived at the barrier.

(C) 2009 Radon Labs GmbH
Timing::CalendarTime
#include <calendartime.h>

Inheritance diagram for Timing::CalendarTime:
Detailed Description

Allows to obtain the current point in time as year, month, day, etc... down to milliseconds, convert between filetime and CalendarTime, and format the time to a human readable string.

(C) 2007 Radon Labs GmbH
Timing::Timer
#include <timer.h>

Inheritance diagram for Timing::Timer:

```
Win32::Win32Timer
    |
    v
Timing::Timer
```
Detailed Description

A timer object is the most basic object for time measurement. More advanced timing classes often build on top of Timer.

(C) 2006 Radon Labs GmbH
Util::Array
Util::Array< TYPE > Class Template Reference

#include <array.h>
Detailed Description

template<class TYPE>
class Util::Array< TYPE >

Nebula3’s dynamic array class. This class is also used by most other collection classes.

The default constructor will not pre-allocate elements, so no space is wasted as long as no elements are added. As soon as the first element is added to the array, an initial buffer of 16 elements is created. Whenever the element buffer would overflow, a new buffer of twice the size of the previous buffer is created and the existing elements are then copied over to the new buffer. The element buffer will never shrink, the only way to reclaim unused memory is to copy the Array to a new Array object. This is usually not a problem since most arrays will oscillate around some specific size, so once the array has reached this specific size, no costly memory free or allocs will be performed.

It is possible to sort the array using the Sort() method, this uses std::sort (one of the very few exceptions where the STL is used in Nebula3).

One should generally be careful with costly copy operators, the Array class (and the other container classes using Array) may do some heavy element shuffling in some situations (especially when sorting and erasing elements).

(C) 2006 RadonLabs GmbH
### Public Types

<table>
<thead>
<tr>
<th>Typedef TYPE *</th>
<th><strong>Iterator</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>define iterator</em></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Array</strong> ()</td>
<td>constructor with default parameters</td>
</tr>
<tr>
<td><strong>Array</strong> (SizeT initialCapacity, SizeT initialGrow)</td>
<td>constructor with initial size and grow size</td>
</tr>
<tr>
<td><strong>Array</strong> (SizeT initialSize, SizeT initialGrow, const TYPE &amp;initialValue)</td>
<td>constructor with initial size, grow size and initial values</td>
</tr>
<tr>
<td><strong>Array</strong> (const Array&lt;TYPE&gt; &amp;rhs)</td>
<td>copy constructor</td>
</tr>
<tr>
<td>~<strong>Array</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void operator= (const Array&lt;TYPE&gt; &amp;rhs)</td>
<td>assignment operator</td>
</tr>
<tr>
<td>TYPE &amp; operator[] (IndexT index) const</td>
<td>[] operator</td>
</tr>
<tr>
<td>bool operator== (const Array&lt;TYPE&gt; &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool operator!= (const Array&lt;TYPE&gt; &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
</tbody>
</table>

```template<typename T>```  
```
T As () const
```

`convert to "anything"`  

void **Append** (const TYPE &elm)  
append element to end of array  

void **AppendArray** (const Array<TYPE> &rhs)  
append the contents of an array to this array  

void **Reserve** (SizeT num)  
inecrease capacity to fit N more elements into the array  

SizeT **Size** () const  
get number of elements in array  

SizeT **Capacity** () const  
get overall allocated size of array in number of elements  

TYPE & **Front** () const  
return reference to first element
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TYPE &amp; Back () const</code></td>
<td>return reference to last element</td>
</tr>
<tr>
<td>bool <code>IsEmpty () const</code></td>
<td>return true if array empty</td>
</tr>
<tr>
<td>void <code>EraseIndex (IndexT index)</code></td>
<td>erase element at index, keep sorting intact</td>
</tr>
<tr>
<td>Iterator <code>Erase (Iterator iter)</code></td>
<td>erase element pointed to by iterator, keep sorting intact</td>
</tr>
<tr>
<td>void <code>EraseIndexSwap (IndexT index)</code></td>
<td>erase element at index, fill gap by swapping in last element, destroys sorting!</td>
</tr>
<tr>
<td>Iterator <code>EraseSwap (Iterator iter)</code></td>
<td>erase element at iterator, fill gap by swapping in last element, destroys sorting!</td>
</tr>
<tr>
<td>void <code>Insert (IndexT index, const TYPE &amp;elm)</code></td>
<td>insert element before element at index</td>
</tr>
<tr>
<td>IndexT <code>InsertSorted (const TYPE &amp;elm)</code></td>
<td>insert element into sorted array, return index where element was included</td>
</tr>
<tr>
<td>IndexT <code>InsertAtEndOfIdenticalRange (IndexT startIndex, const TYPE &amp;elm)</code></td>
<td>insert element at the first non-identical position, return index of inclusion position</td>
</tr>
<tr>
<td>bool <code>IsSorted () const</code></td>
<td>test if the array is sorted, this is a slow operation!</td>
</tr>
<tr>
<td>void <code>Clear ()</code></td>
<td>clear array (calls destructors)</td>
</tr>
<tr>
<td>void <code>Reset ()</code></td>
<td>reset array (does NOT call destructors)</td>
</tr>
<tr>
<td>Iterator <code>Begin () const</code></td>
<td>return iterator to beginning of array</td>
</tr>
<tr>
<td>Iterator <code>End () const</code></td>
<td>return iterator to end of array</td>
</tr>
<tr>
<td>Iterator <code>Find (const TYPE &amp;elm) const</code></td>
<td>find identical element in array, return iterator</td>
</tr>
<tr>
<td>IndexT <code>FindIndex (const TYPE &amp;elm) const</code></td>
<td>find identical element in array, return index, InvalidIndex if not found</td>
</tr>
<tr>
<td>void <code>Fill (IndexT first, SizeT num, const TYPE &amp;elm)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void Realloc (SizeT capacity, SizeT grow)</code></td>
<td>clear contents and preallocate with new attributes</td>
</tr>
<tr>
<td><code>Array&lt;TYPE&gt; Difference (const Array&lt;TYPE&gt; &amp;rhs)</code></td>
<td>returns new array with elements which are not in rhs (slow!)</td>
</tr>
<tr>
<td><code>void Sort ()</code></td>
<td>sort the array</td>
</tr>
<tr>
<td><code>IndexT BinarySearchIndex (const TYPE &amp;elm)</code></td>
<td>do a binary search, requires a sorted array</td>
</tr>
</tbody>
</table>
Member Function Documentation

template<class TYPE>
TYPE &
Util::Array<TYPE< ( IndexT index ) const
> :: operator []

[] operator

Access an element. This method will NOT grow the array, and instead do a range check, which may throw an assertion.

template<class TYPE>
bool const
Util::Array< TYPE< Array< TYPE rhs ) const
> :: operator == > &

equality operator

The equality operator returns true if all elements are identical. The TYPE class must support the equality operator.

template<class TYPE>
bool const
Util::Array< TYPE< Array< TYPE rhs ) const
> :: operator != > &

inequality operator

The inequality operator returns true if at least one element in the array is different, or the array sizes are different.

template<class TYPE>
void
Util::Array< TYPE< SizeT num )
> :: Reserve

increase capacity to fit N more elements into the array
This increases the capacity to make room for N elements. If the number of elements is known before appending the elements, this method can be used to prevent reallocation. If there is already enough room for N more elements, nothing will happen.

NOTE: the functionality of this method has been changed as of 26-Apr-08, it will now only change the capacity of the array, not its size.

template<class TYPE>
void Util::Array<TYPE>::EraseIndexSwap(IndexT index)

erase element at index, fill gap by swapping in last element, destroys sorting!

NOTE: this method is fast but destroys the sorting order!

template<class TYPE>
IndexT Util::Array<TYPE>::InsertSorted(const TYPE &elm)

insert element into sorted array, return index where element was included

This inserts the element into a sorted array. Returns the index at which the element was inserted.

template<class TYPE>
IndexT Util::Array<TYPE>::InsertAtEndOfIdenticalRange(IndexT startIndex, const TYPE &elm)

insert element at the first non-identical position, return index of inclusion position

This inserts an element at the end of a range of identical elements
starting at a given index. Performance is O(n). Returns the index at which the element was added.

template<class TYPE>
bool
Util::Array< TYPE >
>::IsSorted

test if the array is sorted, this is a slow operation!

This tests, whether the array is sorted. This is a slow operation O(n).

template<class TYPE>
void
Util::Array< TYPE >
>::Clear

clear array (calls destructors)

The current implementation of this method does not shrink the preallocated space. It simply sets the array size to 0.

template<class TYPE>
void
Util::Array< TYPE >
>::Reset

reset array (does NOT call destructors)

This is identical with Clear(), but does NOT call destructors (it just resets the size member. USE WITH CARE!

template<class TYPE>
Array< TYPE >
>::Iterator
const
Util::Array< TYPE >
>::Find

find identical element in array, return iterator
Find element in array, return iterator, or 0 if element not found.

**Parameters:**

- `elm` element to find

**Returns:**

- element iterator, or 0 if not found

```cpp
template<class TYPE>
IndexT Util::Array< TYPE >::FindIndex(
  const TYPE& elm)
```

find identical element in array, return index, InvalidIndex if not found

Find element in array, return element index, or InvalidIndex if element not found.

**Parameters:**

- `elm` element to find

**Returns:**

- index to element, or InvalidIndex if not found

```cpp
template<class TYPE>
void Util::Array< TYPE >::Fill(
  IndexT first,
  SizeT num,
  const TYPE& elm)
```

fill array range with element

Fills an array range with the given element value. Will grow the array if necessary

**Parameters:**

- `first` index of first element to start fill
template<class TYPE>
Array<
  TYPE>
const
Util::Array<
  Array<
    TYPE>
rhs
>
>
::Difference

returns new array with elements which are not in rhs (slow!)

Returns a new array with all element which are in rhs, but not in this. Carefull, this method may be very slow with large arrays!

Todo:
this method is broken, check test case to see why!

template<class TYPE>
void
Util::Array<
  TYPE
>
>
::Sort

sort the array

Sorts the array. This just calls the STL sort algorithm.

template<class TYPE>
IndexT Util::Array<
  TYPE
>
>
::BinarySearchIndex
	elm
	&

do a binary search, requires a sorted array

Does a binary search on the array, returns the index of the identical element, or InvalidIndex if not found
Util::BitField
Util::BitField< NUMBITS > Class Template Reference

#include <bitfield.h>
Detailed Description

template<unsigned int NUMBITS>
class Util::BitField< NUMBITS >

Implements large bit field with multiple of 32 bits.

(C) 2009 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BitField ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>BitField (const BitField&lt; NUMBITS &gt; &amp;rhs)</strong></td>
<td>copy constructor</td>
</tr>
<tr>
<td><strong>void operator= (const BitField&lt; NUMBITS &gt; &amp;rhs)</strong></td>
<td>assignment operator</td>
</tr>
<tr>
<td><strong>bool operator== (const BitField&lt; NUMBITS &gt; &amp;rhs)</strong></td>
<td>equality operator</td>
</tr>
<tr>
<td><strong>bool operator!= (const BitField&lt; NUMBITS &gt; &amp;rhs)</strong></td>
<td>inequality operator</td>
</tr>
<tr>
<td><strong>void Clear ()</strong></td>
<td>clear content</td>
</tr>
<tr>
<td><strong>bool IsNull () const</strong></td>
<td>return true if all bits are 0</td>
</tr>
<tr>
<td><strong>void SetBit (IndexT bitIndex)</strong></td>
<td>set a bit by index</td>
</tr>
<tr>
<td><strong>void ClearBit (IndexT bitIndex)</strong></td>
<td>clear a bit by index</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static BitField&lt;NUMBITS&gt; Or (const BitField&lt;NUMBITS&gt; &amp;b0, const BitField&lt;NUMBITS&gt; &amp;b1)</code></td>
<td>set bitfield to OR combination</td>
</tr>
<tr>
<td><code>static BitField&lt;NUMBITS&gt; And (const BitField&lt;NUMBITS&gt; &amp;b0, const BitField&lt;NUMBITS&gt; &amp;b1)</code></td>
<td>set bitfield to AND combination</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Util::Blob
Util::Blob Class Reference

#include <blob.h>
Detailed Description

The **Util::Blob** class encapsulates a chunk of raw memory into a C++ object which can be copied, compared and hashed.

(C) 2006 Radon Labs GmbH
Public Member Functions

```cpp
void * operator new (size_t s) override
new operator

void operator delete (void *ptr) override
delete operator

Blob ()
default constructor

Blob (const void *ptr, SizeT size)
constructor

Blob (SizeT size)
reserve N bytes

Blob (const Blob &rhs)
copy constructor

~Blob ()
destructor

void operator= (const Blob &rhs)
assignment operator

bool operator== (const Blob &rhs) const
equality operator

bool operator!= (const Blob &rhs) const
inequality operator

bool operator> (const Blob &rhs) const
greater operator

bool operator< (const Blob &rhs) const
less operator

bool operator>= (const Blob &rhs) const
greater-equal operator

bool operator<= (const Blob &rhs) const
less-equal operator

bool IsValid () const
return true if the blob contains data

void Reserve (SizeT size)
reserve N bytes

void Trim (SizeT size)
```
trim the size member (without re-allocating!)

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>Set</code> (const void *ptr, SizeT size)</td>
<td>set blob contents</td>
</tr>
<tr>
<td>void *</td>
<td><code>GetPtr</code> () const</td>
<td>get blob ptr</td>
</tr>
<tr>
<td>SizeT</td>
<td><code>Size</code> () const</td>
<td>get blob size</td>
</tr>
<tr>
<td>IndexT</td>
<td><code>HashCode</code> () const</td>
<td>get a hash code (compatible with <code>Util::HashTable</code>)</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Setup()</code></td>
<td>Static Setup method, called by SysFunc::Setup()</td>
</tr>
<tr>
<td><code>Shutdown()</code></td>
<td>Static Shutdown method called by SysFunc::Exit</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Util::CommandLineArgs
#include <commandlineargs.h>
Detailed Description

A universal cmd line argument parser. The command line string must have the form

cmd arg0[=]value0 arg1[=]value1 arg2[=]value2

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CommandLineArgs()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>CommandLineArgs(const String &amp;cmdLine)</code></td>
<td>construct from Win32-style command string</td>
</tr>
<tr>
<td><code>CommandLineArgs(int argc, const char **argv)</code></td>
<td>construct from posix style command string</td>
</tr>
<tr>
<td><code>const String &amp; GetCmdName()</code></td>
<td>get the command name</td>
</tr>
<tr>
<td><code>bool HasArg(const String &amp;arg)</code></td>
<td>return true if arg exists</td>
</tr>
<tr>
<td><code>const String &amp; GetString(const String &amp;name, const String &amp;defaultValue=&quot;&quot;)</code></td>
<td>get string value</td>
</tr>
<tr>
<td><code>int GetInt(const String &amp;name, int defaultValue=0)</code></td>
<td>get int value</td>
</tr>
<tr>
<td><code>float GetFloat(const String &amp;name, float defaultValue=0.0f)</code></td>
<td>get float value</td>
</tr>
<tr>
<td><code>bool GetBool(const String &amp;name, bool defaultValue=false)</code></td>
<td>get bool value (key=value)</td>
</tr>
<tr>
<td><code>bool GetBoolFlag(const String &amp;name)</code></td>
<td>get bool flag (args only needs to exist to trigger as true)</td>
</tr>
<tr>
<td><code>Math::float4 GetFloat4(const String &amp;name, const Math::float4 &amp;defaultValue= Math::float4())</code></td>
<td>get float4 value</td>
</tr>
<tr>
<td><code>Math::matrix44 GetMatrix44(const String &amp;name, const Math::matrix44 &amp;defaultValue= Math::matrix44())</code></td>
<td>get matrix44 value</td>
</tr>
<tr>
<td><code>SizeT GetNumArgs()</code></td>
<td>get number of arguments (excluding command name)</td>
</tr>
<tr>
<td><code>const String &amp; GetStringAtIndex(IndexT index)</code></td>
<td>get string value at index</td>
</tr>
<tr>
<td>Type</td>
<td>Function Name</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>int</td>
<td><code>GetIntAtIndex</code></td>
</tr>
<tr>
<td>float</td>
<td><code>GetFloatAtIndex</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>GetBoolAtIndex</code></td>
</tr>
<tr>
<td><code>Math::float4</code></td>
<td><code>GetFloat4AtIndex</code></td>
</tr>
<tr>
<td><code>Math::matrix44</code></td>
<td><code>GetMatrix44AtIndex</code></td>
</tr>
<tr>
<td>void</td>
<td><code>AppendCommandString</code></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
const String & Util::CommandLineArgs::GetCmdName() const
```

get the command name

Returns the command name.

```cpp
bool const Util::CommandLineArgs::HasArg(String name) const
```

return true if arg exists

Returns true if argument exists.
Util::Crc
Util::Crc Class Reference

#include <crc.h>
Detailed Description

Compute CRC checksums over a range of memory.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Crc</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td><strong>Begin</strong> ()</td>
<td>begin computing a checksum</td>
</tr>
<tr>
<td><strong>Compute</strong> (unsigned char *buf, unsigned int numBytes)</td>
<td>continue computing checksum</td>
</tr>
<tr>
<td><strong>End</strong> ()</td>
<td>finish computing the checksum</td>
</tr>
<tr>
<td><strong>GetResult</strong> () const</td>
<td>get result</td>
</tr>
</tbody>
</table>
void
Util::Crc::Begin()

begin computing a checksum

Begin computing a CRC checksum over several chunks of data. This can be done in multiple runs, which is the only useful way to compute checksum for large files.

void
Util::Crc::Compute(unsigned char *buf,
                   unsigned int numBytes)

continue computing checksum

Do one run of checksum computation for a chunk of data. Must be executed inside Begin()/End().

void
Util::Crc::End()

finish computing the checksum

End checksum computation. This validates the result, so that it can be accessed with GetResult().

unsigned int
Util::Crc::GetResult() const

get result

Get the result of the checksum computation. Must be executed after End().
Util::Delegate
Util::Delegate< ARGTYPE > Class Template Reference

#include <delegate.h>
Detailed Description

template<class ARGTYPE>
class Util::Delegate< ARGTYPE >

Nebula3 delegate class, allows to store a method call into a C++ object for later execution.


(C) 2008 Radon Labs GmbH
Public Member Functions

| Delegate () |
| constructor |
| void operator() (ARGTYPE arg) const |
| invokation operator |
Static Public Member Functions

template<class CLASS, void(CLASS::*)(ARGTYPE) METHOD>
static Delegate< ARGTYPE > FromMethod (CLASS *objPtr)
  setup a new delegate from a method call

template<void(*)(ARGTYPE) FUNCTION>
static Delegate< ARGTYPE > FromFunction ()
  setup a new delegate from a function call
Util::Dictionary
Util::Dictionary< KEYTYPE, VALUETYPE > Class Template Reference

#include <dictionary.h>
Detailed Description

template<class KEYTYPE, class VALUETYPE>
class Util::Dictionary< KEYTYPE, VALUETYPE >

A collection of key/value pairs with quick value retrieval by key at roughly $O(\log n)$.

Internally the dictionary is implemented as a sorted array.

On insertion performance: Key/value pairs are inserted with the Add() method, which normally calls the Util::Array::InsertSorted() method internally. If many insertions are performed at once, it may be beneficial to call BeginBulkAdd() before, and EndBulkAdd() after adding the key/value pairs. Between BeginBulkAdd() and EndBulkAdd(), the Add() method will just append the new elements to the internal array, and only call Util::Array::Sort() inside EndBulkAdd().

Any methods which require the internal array to be sorted will throw an assertion between BeginBulkAdd() and EndBulkAdd().

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dictionary ()</strong></td>
<td>Default constructor</td>
</tr>
<tr>
<td><strong>Dictionary (const Dictionary&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</strong></td>
<td>Copy constructor</td>
</tr>
<tr>
<td><strong>void operator= (const Dictionary&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</strong></td>
<td>Assignment operator</td>
</tr>
<tr>
<td><strong>VALUETYPE &amp; operator[] (const KEYTYPE &amp;key)</strong></td>
<td>Read/write [] operator</td>
</tr>
<tr>
<td><strong>const VALUETYPE &amp; operator[] (const KEYTYPE &amp;key)</strong></td>
<td>Read-only [] operator</td>
</tr>
<tr>
<td><strong>SizeT Size () const</strong></td>
<td>Return number of key/value pairs in the dictionary</td>
</tr>
<tr>
<td><strong>void Clear ()</strong></td>
<td>Clear the dictionary</td>
</tr>
<tr>
<td><strong>bool IsEmpty () const</strong></td>
<td>Return true if empty</td>
</tr>
<tr>
<td><strong>void Reserve (SizeT numElements)</strong></td>
<td>Reserve space (useful if number of elements is known beforehand)</td>
</tr>
<tr>
<td><strong>void BeginBulkAdd ()</strong></td>
<td>Begin a bulk insert (array will be sorted at End)</td>
</tr>
<tr>
<td><strong>void Add (const KeyValuePair&lt;KEYTYPE, VALUETYPE&gt; &amp;kvp)</strong></td>
<td>Add a key/value pair</td>
</tr>
<tr>
<td><strong>void Add (const KEYTYPE &amp;key, const VALUETYPE &amp;value)</strong></td>
<td>Add a key and associated value</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>EndBulkAdd()</code></td>
<td>end a bulk insert (this will sort the internal array)</td>
</tr>
<tr>
<td>void <code>Erase</code> (const KEYTYPE &amp;key)</td>
<td>erase a key and its associated value</td>
</tr>
<tr>
<td>void <code>EraseAtIndex</code> (IndexT index)</td>
<td>erase a key at index</td>
</tr>
<tr>
<td>IndexT <code>FindIndex</code> (const KEYTYPE &amp;key) const</td>
<td>find index of key/value pair (InvalidIndex if doesn't exist)</td>
</tr>
<tr>
<td>bool <code>Contains</code> (const KEYTYPE &amp;key) const</td>
<td>return true if key exists in the array</td>
</tr>
<tr>
<td>const KEYTYPE &amp; <code>KeyAtIndex</code> (IndexT index) const</td>
<td>get a key at given index</td>
</tr>
<tr>
<td>VALUETYPE &amp; <code>ValueAtIndex</code> (IndexT index) const</td>
<td>access to value at given index</td>
</tr>
<tr>
<td>const VALUETYPE &amp; <code>ValueAtIndex</code> (IndexT index) const</td>
<td>get a value at given index</td>
</tr>
<tr>
<td><code>KeyValuePair&lt;KEYTYPE, VALUETYPE&gt; &amp;</code> <code>KeyValuePairAtIndex</code> (IndexT index) const</td>
<td>get key/value pair at index</td>
</tr>
<tr>
<td><code>Array&lt;KEYTYPE&gt;</code> &gt; <code>KeysAsArray</code> () const</td>
<td>get all keys as an <code>Util::Array</code></td>
</tr>
<tr>
<td><code>Array&lt;VALUETYPE&gt;</code> &gt; <code>ValuesAsArray</code> () const</td>
<td>get all keys as an <code>Util::Array</code></td>
</tr>
<tr>
<td>template&lt;class RETURNTYPE&gt; <code>KeysAs</code> () const</td>
<td>get all keys as (typically) an array</td>
</tr>
<tr>
<td>template&lt;class RETURNTYPE&gt; <code>ValuesAs</code> () const</td>
<td>get all keys as (typically) an array</td>
</tr>
</tbody>
</table>
Protected Member Functions

void SortIfDirty () const
make sure the key value pair array is sorted
Util::FixedArray
Util::FixedArray< TYPE > Class Template Reference

#include <fixedarray.h>
Detailed Description

template<class TYPE>
class Util::FixedArray< TYPE >

Implements a fixed size one-dimensional array.

(C) 2006 Radon Labs GmbH
## Public Types

typedef TYPE * **Iterator**

*define element iterator*
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FixedArray()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>FixedArray(SizeT s)</code></td>
<td>constructor with size</td>
</tr>
<tr>
<td><code>FixedArray(SizeT s, const TYPE &amp;initialValue)</code></td>
<td>constructor with size and initial value</td>
</tr>
<tr>
<td><code>FixedArray(const FixedArray&lt;TYPE&gt; &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>~FixedArray()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void operator=(const FixedArray&lt;TYPE&gt; &amp;rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td><code>TYPE &amp; operator[](IndexT index)</code></td>
<td>write [] operator</td>
</tr>
<tr>
<td><code>bool operator==(const FixedArray&lt;TYPE&gt; &amp;rhs)</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const FixedArray&lt;TYPE&gt; &amp;rhs)</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>void SetSize(SizeT s)</code></td>
<td>set number of elements (clears existing content)</td>
</tr>
<tr>
<td><code>SizeT Size()</code></td>
<td>get number of elements</td>
</tr>
<tr>
<td><code>void Resize(SizeT newSize)</code></td>
<td>resize array without deleting existing content</td>
</tr>
<tr>
<td><code>bool IsEmpty()</code></td>
<td>return true if array if empty (has no elements)</td>
</tr>
<tr>
<td><code>void Clear()</code></td>
<td>clear the array, free elements</td>
</tr>
<tr>
<td><code>void Fill(const TYPE &amp;val)</code></td>
<td>fill the entire array with a value</td>
</tr>
<tr>
<td><code>void Fill(IndexT first, SizeT num, const TYPE &amp;val)</code></td>
<td>fill array range with element</td>
</tr>
<tr>
<td><code>Iterator Begin()</code></td>
<td>const</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Iterator</strong></td>
<td></td>
</tr>
<tr>
<td><code>End () const</code></td>
<td>get iterator past last element</td>
</tr>
<tr>
<td><strong>Iterator</strong></td>
<td></td>
</tr>
<tr>
<td><code>Find (const TYPE &amp;val) const</code></td>
<td>find identical element in unsorted array (slow)</td>
</tr>
<tr>
<td><strong>IndexT</strong></td>
<td></td>
</tr>
<tr>
<td><code>FindIndex (const TYPE &amp;val) const</code></td>
<td>find index of identical element in unsorted array (slow)</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td></td>
</tr>
<tr>
<td><code>Sort ()</code></td>
<td>sort the array</td>
</tr>
<tr>
<td><strong>IndexT</strong></td>
<td></td>
</tr>
<tr>
<td><code>BinarySearchIndex (const TYPE &amp;val) const</code></td>
<td>do a binary search, requires a sorted array</td>
</tr>
<tr>
<td><strong>Array&lt; TYPE &gt;</strong></td>
<td></td>
</tr>
<tr>
<td><code>AsArray () const</code></td>
<td>return content as <code>Array</code> (slow)</td>
</tr>
</tbody>
</table>
Member Function Documentation

template<class TYPE>
IndexT
Util::FixedArray< TYPE >::BinarySearchIndex

const TYPE&

const ( TYPE elm ) const

&

do a binary search, requires a sorted array

Todo:
    hmm, this is copy-pasted from Array...

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Util::FixedTable
Util::FixedTable< TYPE > Class
Template Reference

#include <fixedtable.h>
Detailed Description

```cpp
template<class TYPE>
class Util::FixedTable< TYPE >
```

A fixed-size 2-dimensional array.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FixedTable()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>FixedTable(SizeT w, SizeT h)</code></td>
<td>constructor with size</td>
</tr>
<tr>
<td><code>FixedTable(SizeT w, SizeT h, const TYPE &amp;val)</code></td>
<td>constructor with size and initialized contents</td>
</tr>
<tr>
<td><code>FixedTable(const FixedTable&lt; TYPE &gt; &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>~FixedTable()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>void <code>operator= (const FixedTable&lt; TYPE &gt; &amp;rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td>bool <code>operator== (const FixedTable&lt; TYPE &gt; &amp;rhs)</code></td>
<td>equality operator</td>
</tr>
<tr>
<td>bool <code>operator!= (const FixedTable&lt; TYPE &gt; &amp;rhs)</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td>void <code>SetSize (SizeT w, SizeT h)</code></td>
<td>set width and height (clears existing content)</td>
</tr>
<tr>
<td><code>SizeT Width () const</code></td>
<td>get width</td>
</tr>
<tr>
<td><code>SizeT Height () const</code></td>
<td>get height</td>
</tr>
<tr>
<td>void <code>Clear (const TYPE &amp;val)</code></td>
<td>clear the table with value</td>
</tr>
<tr>
<td>void <code>Set (IndexT x, IndexT y, const TYPE &amp;val)</code></td>
<td>set value at $[x,y]$ position</td>
</tr>
<tr>
<td><code>TYPE &amp; At (IndexT x, IndexT y)</code></td>
<td>access value at $[x,y]$ position</td>
</tr>
</tbody>
</table>
Util::FourCC
Util::FourCC Class Reference

#include <fourcc.h>
Detailed Description

A four-character-code is a quasi-human-readable 32-bit-id.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FourCC ()</td>
<td>default constructor</td>
</tr>
<tr>
<td>FourCC (uint f)</td>
<td>construct from 32-bit-value (e.g. FourCC('ABCD'))</td>
</tr>
<tr>
<td>FourCC (const String &amp;s)</td>
<td>construct from string</td>
</tr>
<tr>
<td>bool operator== (const FourCC &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool operator!=(const FourCC &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td>bool operator&lt; (const FourCC &amp;rhs) const</td>
<td>less-than operator</td>
</tr>
<tr>
<td>bool operator&lt;= (const FourCC &amp;rhs) const</td>
<td>less-or-equal operator</td>
</tr>
<tr>
<td>bool operator&gt; (const FourCC &amp;rhs) const</td>
<td>greater-than operator</td>
</tr>
<tr>
<td>bool operator&gt;= (const FourCC &amp;rhs) const</td>
<td>greater-or-equal operator</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>return true if valid</td>
</tr>
<tr>
<td>void SetFromUInt (uint f)</td>
<td>set from 32-bit-value</td>
</tr>
<tr>
<td>uint AsUInt () const</td>
<td>get as 32-bit-value</td>
</tr>
<tr>
<td>void SetFromString (const String &amp;s)</td>
<td>set as string</td>
</tr>
<tr>
<td>String AsString () const</td>
<td>get as string</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String <strong>ToString</strong> (const FourCC &amp;f)</td>
<td>convert fourcc to string</td>
</tr>
<tr>
<td>static FourCC <strong>FromString</strong> (const String &amp;s)</td>
<td>convert string to fourcc</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Util::GlobalStringAtomTable
Util::GlobalStringAtomTable Class Reference

#include <globalstringatomtable.h>

Inheritance diagram for Util::GlobalStringAtomTable:
Detailed Description

Global string atom table. This is the definitive string atom table which contains the string of all string atoms of all threads. Locking is necessary to lookup or add a string (that's why thread-local string atom tables exist as local cache to prevent too much locking of the global table).

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GlobalStringAtomTable ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~GlobalStringAtomTable ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>StringBuffer * <strong>GetGlobalStringBuffer ()</strong> const</td>
<td>get pointer to global string buffer</td>
</tr>
<tr>
<td>DebugInfo <strong>GetDebugInfo ()</strong> const</td>
<td>debug functionality: get copy of the string atom table</td>
</tr>
</tbody>
</table>
Protected Member Functions

const char * Find (const char *str) const

find a string pointer in the atom table
Data Structures

```go
struct DebugInfo
debg functionality: DebugInfo struct More...
```
Member Function Documentation

GlobalStringAtomTable::DebugInfo
Util::GlobalStringAtomTable::GetDebugInfo ( ) const

ddebug functionality: get copy of the string atom table

Debug method: get an array with all string in the table.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Util::GlobalStringAtomTable::DebugInfo
Util::GlobalStringAtomTable::DebugInfo

Struct Reference

#include <globalstringatomtable.h>
Detailed Description

detail functionality: **DebugInfo** struct
Util::Guid
Util::Guid Class Reference

#include <guid.h>
Detailed Description

Implements a GUID.

(C) 2006 Radon Labs GmbH
Util::HashTable
Util::HashTable< KEYTYPE,
VALUETYPE > Class Template Reference

#include <hashtable.h>
Detailed Description

```template<class KEYTYPE, class VALUETYPE>
class Util::HashTable< KEYTYPE, VALUETYPE >
```

Organizes key/value pairs by a hash code. Looks very similar to a Dictionary, but may provide better search times (up to $O(1)$) by computing a (ideally unique) hash code on the key and using that as an index into an array. The flipside is that the key class must provide a hash code and the memory footprint may be larger than Dictionary.

The default capacity is 128. Matching the capacity against the number of expected elements in the hash table is one key to get optimal insertion and search times, the other is to provide a good (and fast) hash code computation which produces as few collisions as possible for the key type.

The key class must implement the following method in order to work with the HashTable: `IndexT HashCode() const;`

The Util::String class implements this method as an example. Internally the hash table is implemented as a fixed array of sorted arrays. The fixed array is indexed by the hash code of the key, the sorted arrays contain all values with identical keys.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HashTable ()</strong></td>
<td>default constructor</td>
</tr>
<tr>
<td><strong>HashTable (SizeT capacity)</strong></td>
<td>constructor with capacity</td>
</tr>
<tr>
<td><strong>HashTable (const HashTable&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</strong></td>
<td>copy constructor</td>
</tr>
<tr>
<td><strong>void operator= (const HashTable&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</strong></td>
<td>assignment operator</td>
</tr>
<tr>
<td><strong>VALUETYPE &amp; operator[] (const KEYTYPE &amp;key) const</strong></td>
<td>read/write [] operator, assertion if key not found</td>
</tr>
<tr>
<td><strong>SizeT Size () const</strong></td>
<td>return current number of values in the hashtable</td>
</tr>
<tr>
<td><strong>SizeT Capacity () const</strong></td>
<td>return fixed-size capacity of the hash table</td>
</tr>
<tr>
<td><strong>void Clear ()</strong></td>
<td>clear the hashtable</td>
</tr>
<tr>
<td><strong>bool IsEmpty () const</strong></td>
<td>return true if empty</td>
</tr>
<tr>
<td><strong>void Add (const KeyValuePair&lt;KEYTYPE, VALUETYPE&gt; &amp;kvp)</strong></td>
<td>add a key/value pair object to the hash table</td>
</tr>
<tr>
<td><strong>void Add (const KEYTYPE &amp;key, const VALUETYPE &amp;value)</strong></td>
<td>add a key and associated value</td>
</tr>
<tr>
<td><strong>void Erase (const KEYTYPE &amp;key)</strong></td>
<td>erase an entry</td>
</tr>
<tr>
<td><strong>bool Contains (const KEYTYPE &amp;key) const</strong></td>
<td>return true if key exists in the array</td>
</tr>
<tr>
<td><strong>Array&lt;KeyValuePair&lt;KEYTYPE, VALUETYPE&gt;&gt; Content () const</strong></td>
<td>return array of all key/value pairs in the table (slow)</td>
</tr>
</tbody>
</table>
Util::KeyValuePair
Util::KeyValuePair< KEYTYPE, VALUETYPE > Class Template Reference

#include <keyvaluepair.h>
Detailed Description

template<class KEYTYPE, class VALUETYPE>
class Util::KeyValuePair< KEYTYPE, VALUETYPE >

Key/Value pair objects are used by most associative container classes, like Dictionary or HashTable.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>KeyValuePair()</code></td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><code>KeyValuePair(const KEYTYPE &amp;k, const VALUETYPE &amp;v)</code></td>
<td><em>constructor with key and value</em></td>
</tr>
<tr>
<td><code>KeyValuePair(const KEYTYPE &amp;k)</code></td>
<td><em>constructor with key and undefined value</em></td>
</tr>
<tr>
<td><code>KeyValuePair(const KeyValuePair&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</code></td>
<td><em>copy constructor</em></td>
</tr>
<tr>
<td><code>void operator=(const KeyValuePair&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</code></td>
<td><em>assignment operator</em></td>
</tr>
<tr>
<td><code>bool operator==(const KeyValuePair&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</code></td>
<td><em>equality operator</em></td>
</tr>
<tr>
<td><code>bool operator!=(const KeyValuePair&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</code></td>
<td><em>inequality operator</em></td>
</tr>
<tr>
<td><code>bool operator&gt;(const KeyValuePair&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</code></td>
<td><em>greater operator</em></td>
</tr>
<tr>
<td><code>bool operator&gt;=(const KeyValuePair&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</code></td>
<td><em>greater-or-equal operator</em></td>
</tr>
<tr>
<td><code>bool operator&lt;(const KeyValuePair&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</code></td>
<td><em>lesser operator</em></td>
</tr>
<tr>
<td><code>bool operator&lt;=(const KeyValuePair&lt;KEYTYPE, VALUETYPE&gt; &amp;rhs)</code></td>
<td><em>lesser-or-equal operator</em></td>
</tr>
<tr>
<td><code>VALUETYPE &amp; Value()</code></td>
<td><em>read/write access to value</em></td>
</tr>
<tr>
<td>const KEYTYPE &amp;</td>
<td>Key () const</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>const VALUETYPE &amp;</td>
<td>Value () const</td>
</tr>
</tbody>
</table>
Constructor & Destructor Documentation

template<class KEYTYPE, class VALUETYPE>

```cpp
Util::KeyValuePair< const
KEYTYPE,
VALUETYPE
>&
>::KeyValuePair

constructor with key and undefined value

This strange constructor is useful for search-by-key if the key-value-pairs are stored in a Util::Array.
```
Util::List
Util::List< TYPE > Class Template Reference

#include <list.h>

Inheritance diagram for Util::List< TYPE >:

![Inheritance Diagram](image.png)
Detailed Description

template<class TYPE>
class Util::List< TYPE >

Implements a doubly linked list. Since list elements can be all over the place in memory, dynamic arrays are often the better choice, unless insert/remove performance is more important then traversal performance.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>List ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>List (const List&lt; TYPE &gt; &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>~List ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void operator= (const List&lt; TYPE &gt; &amp;rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td><code>bool IsEmpty () const</code></td>
<td>return true if the list is empty</td>
</tr>
<tr>
<td><code>SizeT Size () const</code></td>
<td>get number of elements in list (slow)</td>
</tr>
<tr>
<td><code>void Clear ()</code></td>
<td>clear list</td>
</tr>
<tr>
<td><code>void AddList (const List&lt; TYPE &gt; &amp;l)</code></td>
<td>add contents of other list to this list</td>
</tr>
<tr>
<td><code>Iterator AddAfter (Iterator iter, const TYPE &amp;e)</code></td>
<td>add element after given element</td>
</tr>
<tr>
<td><code>Iterator AddBefore (Iterator iter, const TYPE &amp;e)</code></td>
<td>add element before given element</td>
</tr>
<tr>
<td><code>Iterator AddFront (const TYPE &amp;e)</code></td>
<td>add element to beginning of list</td>
</tr>
<tr>
<td><code>Iterator AddBack (const TYPE &amp;e)</code></td>
<td>add element to end of list</td>
</tr>
<tr>
<td><code>TYPE RemoveFront ()</code></td>
<td>remove first element of list</td>
</tr>
<tr>
<td><code>TYPE RemoveBack ()</code></td>
<td>remove last element of list</td>
</tr>
<tr>
<td><code>TYPE Remove (Iterator iter)</code></td>
<td>remove given element</td>
</tr>
<tr>
<td><code>TYPE &amp; Front () const</code></td>
<td>get first element</td>
</tr>
<tr>
<td><code>TYPE &amp; Back () const</code></td>
<td></td>
</tr>
<tr>
<td>Iterator</td>
<td>Begin () const</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>Iterator</td>
<td>End () const</td>
</tr>
<tr>
<td>Iterator</td>
<td>Find (const TYPE &amp;e, Iterator start) const</td>
</tr>
</tbody>
</table>
Data Structures

class Iterator

the list iterator More...
Util::List::Iterator
Util::List< TYPE >::Iterator Class Reference

#include <list.h>
Detailed Description

template<class TYPE>
class Util::List<TYPE>::Iterator

the list iterator
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Iterator()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>Iterator(Node *node)</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>Iterator(const Iterator &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>Iterator &amp;</code></td>
<td><code>operator=</code> (const Iterator &amp;rhs)</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>operator==</code> (const Iterator &amp;rhs) const</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>operator!=</code> (const Iterator &amp;rhs) const</td>
</tr>
<tr>
<td><code>const Iterator &amp;</code></td>
<td><code>operator++()</code></td>
</tr>
<tr>
<td><code>const Iterator &amp;</code></td>
<td><code>operator++()</code> (int)</td>
</tr>
<tr>
<td><code>const Iterator &amp;</code></td>
<td><code>operator--()</code></td>
</tr>
<tr>
<td><code>const Iterator &amp;</code></td>
<td><code>operator--()</code> (int)</td>
</tr>
<tr>
<td><code>operator bool()</code> const</td>
<td>bool operator</td>
</tr>
<tr>
<td><code>TYPE *</code></td>
<td><code>operator-&gt;()</code> const</td>
</tr>
<tr>
<td><code>TYPE &amp;</code></td>
<td><code>operator *()</code> const</td>
</tr>
</tbody>
</table>
Util::LocalStringAtomTable
Util::LocalStringAtomTable Class Reference

#include <localstringatomtable.h>

Inheritance diagram for Util::LocalStringAtomTable:

Util::StringAtomTableBase

Util::LocalStringAtomTable
Detailed Description

Implements a thread-local string atom table which is used as a cache to prevent excessive locking when creating string atoms.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LocalStringAtomTable()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~LocalStringAtomTable()</code></td>
<td>destructor</td>
</tr>
</tbody>
</table>
Protected Member Functions

const char * Find (const char *str) const
find a string pointer in the atom table
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Util::QuadTree::Node
Util::QuadTree< TYPE >::Node Class Reference

#include <quadtree.h>
Detailed Description

template<class TYPE>
class Util::QuadTree< TYPE >::Node

node in quad tree
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~Node ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>void Setup (QuadTree&lt; TYPE &gt; <em>tree, uchar _level, ushort _col, ushort _row)</em></strong>*</td>
<td>Recursively initialize the node</td>
</tr>
<tr>
<td><strong>char Level () const</strong></td>
<td>Get the node's level</td>
</tr>
<tr>
<td><strong>ushort Column () const</strong></td>
<td>Get the node's column</td>
</tr>
<tr>
<td><strong>ushort Row () const</strong></td>
<td>Get the node's row</td>
</tr>
<tr>
<td><strong>const Math::bbox &amp; GetBoundingBox () const</strong></td>
<td>Compute the node's bounding box</td>
</tr>
<tr>
<td><strong>Node * FindContainmentNode (const Math::bbox &amp;box)</strong></td>
<td>Recursively find the smallest child node which contains the bounding box</td>
</tr>
<tr>
<td><strong>void SetElement (const TYPE &amp;elm)</strong></td>
<td>Set data element associated with node</td>
</tr>
<tr>
<td><strong>const TYPE &amp; GetElement () const</strong></td>
<td>Get data element</td>
</tr>
<tr>
<td><strong>Node * GetChildAt (IndexT i)</strong></td>
<td>Get child at index</td>
</tr>
</tbody>
</table>
template<class TYPE>
void
Util::QuadTree<Type>::Node::Setup(QuadTree<Type> * tree, 
uchar _level, 
ushort _col, 
ushort _row)

recursively initialize the node

Recursively initialize a quad tree node.

template<class TYPE>
QuadTree<Type>::Node * const
Util::QuadTree<Type>::Node::FindContainmentNode(const Math::bbox checkBox)

recursively find the smallest child node which contains the bounding box

This finds the smallest child node which completely contains the given bounding box. Calls itself recursively.
Util::Queue
Util::Queue< TYPE > Class Template Reference

#include <queue.h>

Inheritance diagram for Util::Queue< TYPE >:
Detailed Description

template<class TYPE>
class Util::Queue< TYPE >

Nebula3’s queue class (a FIFO container).

(C) 2006 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Queue (const Queue&lt; TYPE &gt; &amp;rhs)</strong></td>
<td>copy constructor</td>
</tr>
<tr>
<td><strong>Queue ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>void operator= (const Queue&lt; TYPE &gt; &amp;rhs)</strong></td>
<td>assignment operator</td>
</tr>
<tr>
<td><strong>TYPE &amp; operator[] (IndexT index) const</strong></td>
<td>access element by index, 0 is the frontmost element (next to be dequeued)</td>
</tr>
<tr>
<td><strong>bool operator== (const Queue&lt; TYPE &gt; &amp;rhs) const</strong></td>
<td>equality operator</td>
</tr>
<tr>
<td><strong>bool operator!= (const Queue&lt; TYPE &gt; &amp;rhs) const</strong></td>
<td>inequality operator</td>
</tr>
<tr>
<td><strong>void Reserve (SizeT num)</strong></td>
<td>increase capacity to fit N more elements into the queue</td>
</tr>
<tr>
<td><strong>SizeT Size () const</strong></td>
<td>returns number of elements in the queue</td>
</tr>
<tr>
<td><strong>bool IsEmpty () const</strong></td>
<td>return true if queue is empty</td>
</tr>
<tr>
<td><strong>void Clear ()</strong></td>
<td>remove all elements from the queue</td>
</tr>
<tr>
<td><strong>bool Contains (const TYPE &amp;e) const</strong></td>
<td>return true if queue contains element</td>
</tr>
<tr>
<td><strong>void Enqueue (const TYPE &amp;e)</strong></td>
<td>add element to the back of the queue</td>
</tr>
<tr>
<td><strong>TYPE Dequeue ()</strong></td>
<td>remove the element from the front of the queue</td>
</tr>
<tr>
<td><strong>TYPE &amp; Peek () const</strong></td>
<td>access to element at front of queue without removing it</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar
Util::RandomNumberTable
Util::RandomNumberTable Class Reference

#include <randomnumbertable.h>
Detailed Description

A table-based random-number generator. Returns the same random number for a given key.

(C) 2008 Radon Labs GmbH
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static float</th>
<th>Rand (IndexT key)</th>
<th>return a pseudo-random number between 0 and 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>static float</td>
<td>Rand (IndexT key, float minVal, float maxVal)</td>
<td>return a pseudo random number between min and max</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Util::RingBuffer
Util::RingBuffer< TYPE > Class Template Reference

#include <ringbuffer.h>
Detailed Description

template<class TYPE>
class Util::RingBuffer< TYPE >

A ring buffer stores up to a maximum number of elements in a circular fashion. If the buffer is full, the newest element overwrites the oldest element.

(C) 2008 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Public Member Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RingBuffer ()</strong></td>
</tr>
<tr>
<td>default constructor</td>
</tr>
<tr>
<td><strong>RingBuffer (SizeT capacity)</strong></td>
</tr>
<tr>
<td>constructor with size</td>
</tr>
<tr>
<td><strong>RingBuffer (const RingBuffer&lt;TYPE&gt; &amp;rhs)</strong></td>
</tr>
<tr>
<td>copy constructor</td>
</tr>
<tr>
<td><strong>~RingBuffer ()</strong></td>
</tr>
<tr>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>operator= (const RingBuffer&lt;TYPE&gt; &amp;rhs)</strong></td>
</tr>
<tr>
<td>assignment operator</td>
</tr>
<tr>
<td>TYPE &amp; <strong>operator[] (IndexT index) const</strong></td>
</tr>
<tr>
<td>index operator</td>
</tr>
<tr>
<td>void <strong>SetCapacity (SizeT newCapacity)</strong></td>
</tr>
<tr>
<td>set capacity (clear previous content)</td>
</tr>
<tr>
<td>SizeT <strong>Capacity () const</strong></td>
</tr>
<tr>
<td>get capacity</td>
</tr>
<tr>
<td>SizeT <strong>Size () const</strong></td>
</tr>
<tr>
<td>get number of elements in ring buffer</td>
</tr>
<tr>
<td>void <strong>Add (const TYPE &amp;elm)</strong></td>
</tr>
<tr>
<td>add an element to the ring buffer</td>
</tr>
<tr>
<td>bool <strong>IsEmpty () const</strong></td>
</tr>
<tr>
<td>return true if ring buffer is empty</td>
</tr>
<tr>
<td>void <strong>Reset ()</strong></td>
</tr>
<tr>
<td>reset ring buffer, just reset the head/base indices without calling destructors</td>
</tr>
<tr>
<td>TYPE &amp; <strong>Front () const</strong></td>
</tr>
<tr>
<td>return reference to first element</td>
</tr>
<tr>
<td>TYPE &amp; <strong>Back () const</strong></td>
</tr>
<tr>
<td>return reference to last element</td>
</tr>
<tr>
<td>Array&lt;TYPE&gt; <strong>AsArray () const</strong></td>
</tr>
<tr>
<td>return all values as array</td>
</tr>
<tr>
<td>TYPE * <strong>GetBuffer ()</strong></td>
</tr>
<tr>
<td>get real linear underlying buffer</td>
</tr>
</tbody>
</table>
const TYPE * GetBuffer () const

get real linear underlying buffer
Util::RunLengthCodec
Util::RunLengthCodec Class Reference

#include <runlengthcodec.h>
Detailed Description

A simple byte-based runlength encoder/decoder. Note that the encoded size may actually be bigger then the original size for random data!

(C) 2008 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static SizeT</td>
<td><strong>GetSafeRLEBufferSize</strong> (SizeT srcBufferSize)</td>
<td>Get a safe size for the destination buffer</td>
</tr>
<tr>
<td>static SizeT</td>
<td><strong>ComputeDecodedSize</strong> (const uchar *srcPtr, SizeT srcNumBytes)</td>
<td>Compute the decoded byte size from an RLE encoded stream</td>
</tr>
<tr>
<td>static SizeT</td>
<td><strong>Encode</strong> (const uchar *srcPtr, SizeT srcNumBytes, uchar *dstPtr, SizeT dstNumBytes)</td>
<td>Encode byte buffer to RLE stream, returns size of RLE stream</td>
</tr>
<tr>
<td>static SizeT</td>
<td><strong>Decode</strong> (const uchar *srcPtr, SizeT srcNumBytes, uchar *dstPtr, SizeT dstNumBytes)</td>
<td>Decode RLE stream to byte buffer</td>
</tr>
</tbody>
</table>
Member Function Documentation

SizeT Util::RunLengthCodec::GetSafeRLEBufferSize (SizeT srcBufferSize) [static]

get a safe size for the destination buffer

Get some safe destination buffer size for runlength-encoding. This actually can be up to twice as big as the source buffer for completely random data!
Util::SimpleTree
Util::SimpleTree< VALUETYPE > Class Template Reference

#include <simpletree.h>
Detailed Description

template<class VALUETYPE>
class Util::SimpleTree< VALUETYPE >

A simple tree class which stores its nodes in Util::Arrays.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleTree ()</td>
<td>default constructor</td>
</tr>
<tr>
<td>Node &amp; Root ()</td>
<td>read/write access to root element</td>
</tr>
<tr>
<td>const Node &amp; Root () const</td>
<td>read-only access to root element</td>
</tr>
</tbody>
</table>
class Node
public node class More...
Util::SimpleTree::Node
Util::SimpleTree< VALUETYPE >::Node
Class Reference

#include <simpletree.h>
Detailed Description

template<class VALUETYPE>
class Util::SimpleTree< VALUETYPE >::Node

public node class
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node ()</strong></td>
<td>default constructor</td>
</tr>
<tr>
<td><strong>Node (const Node &amp;parent, const VALUETYPE &amp;val)</strong></td>
<td>constructor with parent and value</td>
</tr>
<tr>
<td><strong>~Node ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td>increment refcount</td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
<td>decrement refcount</td>
</tr>
<tr>
<td>const Node &amp; <strong>operator[](IndexT i) const</strong></td>
<td>get read-only child by index</td>
</tr>
<tr>
<td>**Node &amp; <strong>operator[](IndexT i)</strong></td>
<td>get read/write child by index</td>
</tr>
<tr>
<td>const Node &amp; <strong>Child (IndexT i) const</strong></td>
<td>get read-only child element at index</td>
</tr>
<tr>
<td>**Node &amp; <strong>Child (IndexT i)</strong></td>
<td>get read/write child element at index</td>
</tr>
<tr>
<td>bool <strong>HasParent () const</strong></td>
<td>return true if the node has a parent</td>
</tr>
<tr>
<td>**Node &amp; **Parent ()</td>
<td>read/write access to parent</td>
</tr>
<tr>
<td>const Node &amp; <strong>Parent () const</strong></td>
<td>read-only access to parent</td>
</tr>
<tr>
<td>void <strong>Clear ()</strong></td>
<td>clear children</td>
</tr>
<tr>
<td>SizeT <strong>Size () const</strong></td>
<td>number of children</td>
</tr>
<tr>
<td>bool <strong>IsEmpty () const</strong></td>
<td>return true if empty</td>
</tr>
<tr>
<td>**Node &amp; <strong>Front () const</strong></td>
<td>return first element</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Node &amp; Back</strong> () const</td>
<td>return last element</td>
</tr>
<tr>
<td>void <strong>Append</strong> (const VALUETYPE &amp;val)</td>
<td>add element at back of array</td>
</tr>
<tr>
<td>void <strong>Erase</strong> (IndexT i)</td>
<td>erase at index</td>
</tr>
<tr>
<td>void <strong>Insert</strong> (IndexT index, const VALUETYPE &amp;val)</td>
<td>insert element before element at index</td>
</tr>
<tr>
<td>IndexT <strong>Find</strong> (const VALUETYPE &amp;val) const</td>
<td>find identical element (slow);</td>
</tr>
<tr>
<td>VALUETYPE &amp; <strong>Value</strong> ()</td>
<td>read/write access to value</td>
</tr>
<tr>
<td>const VALUETYPE &amp; <strong>Value</strong> () const</td>
<td>read-only access to value</td>
</tr>
</tbody>
</table>
Util::SparseTable
Util::SparseTable< TYPE > Class
Template Reference

#include <sparsetable.h>
Detailed Description

template<class TYPE>
class Util::SparseTable< TYPE >

A 2D sparse table where many entries may be redundant and support
for multiple entries per cell.

(C) 2009 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SparseTable ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>BeginSetup</strong> (const Array&lt; StringAtom &gt; &amp;columnNames, const Array&lt; StringAtom &gt; &amp;rowNames, SizeT numUnique=0)</td>
<td>setup the sparse table</td>
</tr>
<tr>
<td><strong>AddSingle</strong> (IndexT colIndex, IndexT rowIndex, const TYPE &amp;elm)</td>
<td>add a single new unique entry</td>
</tr>
<tr>
<td><strong>AddMultiple</strong> (IndexT colIndex, IndexT rowIndex, const TYPE *firstElm, SizeT numElements)</td>
<td>add a new multiple entry</td>
</tr>
<tr>
<td><strong>AddReference</strong> (IndexT colIndex, IndexT rowIndex, IndexT refColIndex, IndexT refRowIndex)</td>
<td>add a reference to another column/index</td>
</tr>
<tr>
<td><strong>SetEntryDirect</strong> (IndexT colIndex, IndexT rowIndex, ushort startIndex, ushort numElements)</td>
<td>add a direct reference using an index into the unique element array</td>
</tr>
<tr>
<td><strong>EndSetup</strong> ()</td>
<td>finish setting up the sparse table</td>
</tr>
<tr>
<td><strong>Clear</strong> ()</td>
<td>clear object</td>
</tr>
<tr>
<td><strong>GetNumColumns</strong> () const</td>
<td>get number of columns in the sparse table</td>
</tr>
<tr>
<td><strong>GetNumRows</strong> () const</td>
<td>get number of rows in the sparse table</td>
</tr>
<tr>
<td><strong>HasColumn</strong> (const StringAtom &amp;colName) const</td>
<td>return true if column exists</td>
</tr>
<tr>
<td><strong>HasRow</strong> (const StringAtom &amp;rowName) const</td>
<td>return true if row exists</td>
</tr>
<tr>
<td><strong>GetColumnIndexByName</strong> (const StringAtom &amp;colName) const</td>
<td>return column index by name</td>
</tr>
<tr>
<td>RETURNS</td>
<td>FUNCTION NAME</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
</tr>
<tr>
<td>IndexT</td>
<td>GetRowIndexByName</td>
</tr>
<tr>
<td>SizeT</td>
<td>GetNumUniqueElements</td>
</tr>
<tr>
<td>const TYPE *</td>
<td>GetElements</td>
</tr>
<tr>
<td>const TYPE *</td>
<td>LookupElements</td>
</tr>
</tbody>
</table>
Member Function Documentation

template<class TYPE>
void
Util::SparseTable<TYPE>::AddReference(IndexT colIndex, IndexT rowIndex, IndexT refColIndex, IndexT refRowIndex)

add a reference to another column/index

NOTE: forward references are not allowed!
Util::Stack
Util::Stack< TYPE > Class Template Reference

#include <stack.h>
Detailed Description

template<class TYPE>
class Util::Stack< TYPE >

Nebula3’s stack class (a FILO container).

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Stack ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>Stack (const Stack&lt;TYPE&gt; &amp;rhs)</code></td>
<td>Copy constructor</td>
</tr>
<tr>
<td><code>void operator= (const Stack&lt;TYPE&gt; &amp;rhs)</code></td>
<td>Assignment operator</td>
</tr>
<tr>
<td><code>TYPE &amp; operator[] (IndexT index) const</code></td>
<td>Access element by index, 0 is the topmost element</td>
</tr>
<tr>
<td><code>bool operator==(const Stack&lt;TYPE&gt; &amp;rhs) const</code></td>
<td>Equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const Stack&lt;TYPE&gt; &amp;rhs) const</code></td>
<td>Inequality operator</td>
</tr>
<tr>
<td><code>SizeT Size () const</code></td>
<td>Returns number of elements on stack</td>
</tr>
<tr>
<td><code>bool isEmpty () const</code></td>
<td>Returns true if stack is empty</td>
</tr>
<tr>
<td><code>void Clear ()</code></td>
<td>Remove all elements from the stack</td>
</tr>
<tr>
<td><code>bool Contains (const TYPE &amp;e) const</code></td>
<td>Return true if stack contains element</td>
</tr>
<tr>
<td><code>void Push (const TYPE &amp;e)</code></td>
<td>Push an element on the stack</td>
</tr>
<tr>
<td><code>TYPE &amp; Peek () const</code></td>
<td>Get reference of topmost element of stack, without removing it</td>
</tr>
<tr>
<td><code>TYPE Pop ()</code></td>
<td>Get topmost element of stack, remove element</td>
</tr>
<tr>
<td>home</td>
<td>namespace list</td>
</tr>
<tr>
<td>------</td>
<td>----------------</td>
</tr>
</tbody>
</table>

- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**Util::String**
Util::String Class Reference

#include <string.h>
Detailed Description

Nebula3's universal string class. An empty string object is always 32 bytes big. The string class tries to avoid costly heap allocations with the following tactics:

- a local embedded buffer is used if the string is short enough
- if a heap buffer must be allocated, care is taken to reuse an existing buffer instead of allocating a new buffer if possible (usually if an assigned string fits into the existing buffer, the buffer is reused and not re-allocated)

Heap allocations are performed through a local heap which should be faster then going through the process heap.

Besides the usual string manipulation methods, the String class also offers methods to convert basic Nebula3 datatypes from and to string, and a group of methods which manipulate filename strings.

(C) 2006 RadonLabs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void * operator new (size_t s)</td>
<td>override new operator</td>
</tr>
<tr>
<td>void operator delete (void *ptr)</td>
<td>override delete operator</td>
</tr>
<tr>
<td>String ()</td>
<td>constructor</td>
</tr>
<tr>
<td>String (const String &amp;rhs)</td>
<td>copy constructor</td>
</tr>
<tr>
<td>String (const char *cStr)</td>
<td>construct from C string</td>
</tr>
<tr>
<td>~String ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void operator= (const String &amp;rhs)</td>
<td>assignment operator</td>
</tr>
<tr>
<td>void operator= (const char *cStr)</td>
<td>assign from const char*</td>
</tr>
<tr>
<td>void operator+= (const String &amp;rhs)</td>
<td>+= operator</td>
</tr>
<tr>
<td>char operator[] (IndexT i) const</td>
<td>read-only index operator</td>
</tr>
<tr>
<td>char &amp; operator[] (IndexT i)</td>
<td>read/write index operator</td>
</tr>
<tr>
<td>void Reserve (SizeT newSize)</td>
<td>reserve internal buffer size to prevent heap allos</td>
</tr>
<tr>
<td>SizeT Length () const</td>
<td>return length of string</td>
</tr>
<tr>
<td>void Clear ()</td>
<td>clear the string</td>
</tr>
<tr>
<td>bool isEmpty () const</td>
<td>return true if string object is empty</td>
</tr>
<tr>
<td>bool isValid () const</td>
<td>return true if string object is not empty</td>
</tr>
<tr>
<td>bool CopyToBuffer (char *buf, SizeT bufSize)</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Append (const String &amp;str)</code></td>
<td>append string</td>
</tr>
<tr>
<td><code>Append (const char *str)</code></td>
<td>append c-string</td>
</tr>
<tr>
<td><code>AppendRange (const char *str, SizeT numChars)</code></td>
<td>append a range of characters</td>
</tr>
<tr>
<td><code>ToLower ()</code></td>
<td>convert string to lower case</td>
</tr>
<tr>
<td><code>ToUpper ()</code></td>
<td>convert string to upper case</td>
</tr>
<tr>
<td><code>FirstCharToUpper ()</code></td>
<td>convert first char of string to upper case</td>
</tr>
<tr>
<td><code>Tokenize (const String &amp;whiteSpace, Array&lt;String&gt; &amp;outTokens) const</code></td>
<td>tokenize string into a provided String array (faster if tokens array can be reused)</td>
</tr>
<tr>
<td><code>Tokenize (const String &amp;whiteSpace, char fence, Array&lt;String&gt; &amp;outTokens) const</code></td>
<td>tokenize string, keep strings within fence characters intact (faster if tokens array can be reused)</td>
</tr>
<tr>
<td><code>Tokenize (const String &amp;whiteSpace, char fence) const</code></td>
<td>tokenize string, keep strings within fence characters intact, SLOW since new array will be constructed</td>
</tr>
<tr>
<td><code>ExtractRange (IndexT fromIndex, SizeT numChars) const</code></td>
<td>extract substring</td>
</tr>
<tr>
<td><code>ExtractToEnd (IndexT fromIndex)</code></td>
<td>extract substring to end of this string</td>
</tr>
<tr>
<td><code>Strip (const String &amp;charSet)</code></td>
<td>terminate string at first occurrence of character in set</td>
</tr>
<tr>
<td><code>FindStringIndex (const String &amp;s, IndexT startIndex=0) const</code></td>
<td>return start index of substring, or InvalidIndex if not found</td>
</tr>
<tr>
<td>IndexT</td>
<td><strong>FindCharIndex</strong> (char c, IndexT startIndex=0) const</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return index of character in string, or <em>InvalidIndex</em> if not found</td>
</tr>
<tr>
<td>void</td>
<td><strong>TerminateAtIndex</strong> (IndexT index)</td>
</tr>
<tr>
<td></td>
<td>terminate string at given index</td>
</tr>
<tr>
<td>bool</td>
<td><strong>ContainsCharFromSet</strong> (const <strong>String</strong> &amp;charSet) const</td>
</tr>
<tr>
<td></td>
<td>returns true if string contains any character from set</td>
</tr>
<tr>
<td>void</td>
<td><strong>TrimLeft</strong> (const <strong>String</strong> &amp;charSet)</td>
</tr>
<tr>
<td></td>
<td>delete characters from charset at left side of string</td>
</tr>
<tr>
<td>void</td>
<td><strong>TrimRight</strong> (const <strong>String</strong> &amp;charSet)</td>
</tr>
<tr>
<td></td>
<td>delete characters from charset at right side of string</td>
</tr>
<tr>
<td>void</td>
<td><strong>Trim</strong> (const <strong>String</strong> &amp;charSet)</td>
</tr>
<tr>
<td></td>
<td>trim characters from charset at both sides of string</td>
</tr>
<tr>
<td>void</td>
<td><strong>SubstituteString</strong> (const <strong>String</strong> &amp;str, const <strong>String</strong> &amp;substStr)</td>
</tr>
<tr>
<td></td>
<td>substitute every occurrence of a string with another string</td>
</tr>
<tr>
<td>void</td>
<td><strong>SubstituteChar</strong> (char c, char subst)</td>
</tr>
<tr>
<td></td>
<td>substitute every occurrence of a character with another character</td>
</tr>
<tr>
<td>void</td>
<td>__cdecl <strong>Format</strong> (const char *fmtString,...)</td>
</tr>
<tr>
<td></td>
<td>format string printf-style</td>
</tr>
<tr>
<td>void</td>
<td>__cdecl <strong>FormatArgList</strong> (const char *fmtString, va_list argList)</td>
</tr>
<tr>
<td></td>
<td>format string printf-style with varargs list</td>
</tr>
<tr>
<td>bool</td>
<td><strong>CheckValidCharSet</strong> (const <strong>String</strong> &amp;charSet) const</td>
</tr>
<tr>
<td></td>
<td>return true if string only contains characters from charSet argument</td>
</tr>
<tr>
<td>void</td>
<td><strong>ReplaceChars</strong> (const <strong>String</strong> &amp;charSet, char replacement)</td>
</tr>
<tr>
<td></td>
<td>replace any charset character within a string with the replacement character</td>
</tr>
<tr>
<td>IndexT</td>
<td><strong>HashCode</strong> () const</td>
</tr>
<tr>
<td></td>
<td>return a 32-bit hash code for the string</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetCharPtr</strong> (const char *s)</td>
</tr>
<tr>
<td></td>
<td>set content to char ptr</td>
</tr>
<tr>
<td>void</td>
<td><strong>Set</strong> (const char *ptr, SizeT length)</td>
</tr>
<tr>
<td></td>
<td>set as char ptr, with explicit length</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetInt</strong> (int val)</td>
</tr>
<tr>
<td></td>
<td>set as int value</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetFloat</strong> (float val)</td>
</tr>
<tr>
<td></td>
<td>set as float value</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>void SetBool (bool val)</code></td>
<td>set as bool value</td>
</tr>
<tr>
<td><code>void SetFloat2 (const Math::float2 &amp;v)</code></td>
<td>set as float2 value</td>
</tr>
<tr>
<td><code>void SetFloat4 (const Math::float4 &amp;v)</code></td>
<td>set as float4 value</td>
</tr>
<tr>
<td><code>void SetMatrix44 (const Math::matrix44 &amp;v)</code></td>
<td>set as matrix44 value</td>
</tr>
<tr>
<td><code>template&lt;typename T&gt; void Set (const T &amp;t)</code></td>
<td>generic setter</td>
</tr>
<tr>
<td><code>void AppendInt (int val)</code></td>
<td>append int value</td>
</tr>
<tr>
<td><code>void AppendFloat (float val)</code></td>
<td>append float value</td>
</tr>
<tr>
<td><code>void AppendBool (bool val)</code></td>
<td>append bool value</td>
</tr>
<tr>
<td><code>void AppendFloat2 (const Math::float2 &amp;v)</code></td>
<td>append float2 value</td>
</tr>
<tr>
<td><code>void AppendFloat4 (const Math::float4 &amp;v)</code></td>
<td>append float4 value</td>
</tr>
<tr>
<td><code>void AppendMatrix44 (const Math::matrix44 &amp;v)</code></td>
<td>append matrix44 value</td>
</tr>
<tr>
<td><code>template&lt;typename T&gt; void Append (const T &amp;t)</code></td>
<td>generic append</td>
</tr>
<tr>
<td><code>const char * AsCharPtr () const</code></td>
<td>return contents as character pointer</td>
</tr>
<tr>
<td><code>const char * Get () const</code></td>
<td>*** OBSOLETE *** only Nebula2 compatibility</td>
</tr>
<tr>
<td><code>int AsInt () const</code></td>
<td>return contents as integer</td>
</tr>
<tr>
<td><code>float AsFloat () const</code></td>
<td>return contents as float</td>
</tr>
<tr>
<td><code>bool AsBool () const</code></td>
<td>return contents as bool</td>
</tr>
<tr>
<td><code>Math::float2 AsFloat2 () const</code></td>
<td>return contents as Math::float2</td>
</tr>
</tbody>
</table>
Math::float4  AsFloat4 () const
return contents as float4

Math::matrix44  AsMatrix44 () const
return contents as matrix44

template<typename T>
    T  As () const
convert to "anything"

bool  IsValidInt () const
return true if the content is a valid integer

bool  IsValidFloat () const
return true if the content is a valid float

bool  IsValidBool () const
return true if the content is a valid bool

bool  IsValidFloat2 () const
return true if the content is a valid float2

bool  IsValidFloat4 () const
return true if the content is a valid float4

bool  IsValidMatrix44 () const
return true if content is a valid matrix44

template<typename T>
    bool  IsValid () const
generic valid checker

String  GetFileExtension () const
get filename extension without dot

bool  CheckFileExtension (const String &ext) const
check file extension

void  ConvertBackslashes ()
convert backslashes to slashes

void  StripFileExtension ()
remove file extension

void  ChangeFileExtension (const Util::String &newExt)
change file extension

String  ExtractFileName () const
extract the part after the last directory separator

String  ExtractLastDirName () const
extract the last directory of the path
<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>ExtractDirName () const</td>
<td>extract the part before the last directory separator</td>
</tr>
<tr>
<td>String</td>
<td>ExtractToLastSlash () const</td>
<td>extract path until last slash</td>
</tr>
<tr>
<td>void</td>
<td>ReplaceIllegalFilenameChars (char replacement)</td>
<td>replace illegal filename characters</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>String</code></td>
<td><strong>Concatenate</strong> (const <code>Array&lt;String&gt;</code> &amp;strArray, const <code>String</code> &amp;whiteSpace)</td>
<td>concatenate array of strings into new string</td>
</tr>
<tr>
<td>static <code>bool</code></td>
<td><strong>MatchPattern</strong> (const <code>String</code> &amp;str, const <code>String</code> &amp;pattern)</td>
<td>pattern matching</td>
</tr>
<tr>
<td>static <code>String</code></td>
<td><strong>FromInt</strong> (int i)</td>
<td>construct a string from an int</td>
</tr>
<tr>
<td>static <code>String</code></td>
<td><strong>FromFloat</strong> (float f)</td>
<td>construct a string from a float</td>
</tr>
<tr>
<td>static <code>String</code></td>
<td><strong>FromBool</strong> (bool b)</td>
<td>construct a string from a bool</td>
</tr>
<tr>
<td>static <code>String</code></td>
<td><strong>FromFloat2</strong> (const <code>Math::float2</code> &amp;v)</td>
<td>construct a string from float2</td>
</tr>
<tr>
<td>static <code>String</code></td>
<td><strong>FromFloat4</strong> (const <code>Math::float4</code> &amp;v)</td>
<td>construct a string from float4</td>
</tr>
<tr>
<td>static <code>String</code></td>
<td><strong>FromMatrix44</strong> (const <code>Math::matrix44</code> &amp;m)</td>
<td>construct a string from matrix44</td>
</tr>
<tr>
<td>template&lt;typename T&gt;</td>
<td>static <code>String</code> <strong>From</strong> (const T &amp;t)</td>
<td>convert from &quot;anything&quot;</td>
</tr>
<tr>
<td>static <code>bool</code></td>
<td><strong>IsDigit</strong> (char c)</td>
<td>test if provided character is a digit (0..9)</td>
</tr>
<tr>
<td>static <code>bool</code></td>
<td><strong>IsAlpha</strong> (char c)</td>
<td>test if provided character is an alphabet character (A..Z, a..z)</td>
</tr>
<tr>
<td>static <code>bool</code></td>
<td><strong>IsAlphaNum</strong> (char c)</td>
<td>test if provided character is an alpha-numeric character (A..Z,a..z,0..9)</td>
</tr>
<tr>
<td>static <code>bool</code></td>
<td><strong>IsLower</strong> (char c)</td>
<td>test if provided character is a lower case character</td>
</tr>
<tr>
<td>static <code>bool</code></td>
<td><strong>IsUpper</strong> (char c)</td>
<td>test if provided character is an upper-case character</td>
</tr>
<tr>
<td>static <code>int</code></td>
<td><strong>StrCmp</strong> (const char *str0, const char *str1)</td>
<td>lowlevel string compare wrapper function</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>static int StrLen (const char *str)</code></td>
<td>lowlevel string length function</td>
<td></td>
</tr>
<tr>
<td><code>static const char * StrChr (const char *str, int c)</code></td>
<td>find character in string</td>
<td></td>
</tr>
<tr>
<td><code>static Dictionary&lt; String, String &gt; ParseKeyValuePairs (const String &amp;str)</code></td>
<td>parse key/value pair string (&quot;key0=value0 key1=value1&quot;)</td>
<td></td>
</tr>
</tbody>
</table>
Friends

```cpp
bool operator==(const String &a, const String &b)
  // equality operator
bool operator==(const String &a, const char *cStr)
  // shortcut equality operator
bool operator==(const char *cStr, const String &a)
  shortcut equality operator
bool operator!=(const String &a, const String &b)
  // inequality operator
bool operator<(const String &a, const String &b)
  // less-than operator
bool operator>(const String &a, const String &b)
  // greater-than operator
bool operator<=(const String &a, const String &b)
  // less-or-equal operator
bool operator>=(const String &a, const String &b)
  // greater-then operator
```
Member Function Documentation

char &
Util::String::operator[](IndexT i) [inline]

read/write index operator

NOTE: unlike the read-only indexer, the terminating 0 is NOT a valid part of the string because it may not be overwritten!!!

void
Util::String::Reserve(SizeT newSize) [inline]

reserve internal buffer size to prevent heap allocs

Reserves internal space to prevent excessive heap re-allocations. If you plan to do many Append() operations this may help alot.

SizeT
Util::String::Tokenize(const String& whiteSpace,
                        Array<String>& outTokens)

Tokenize string into a provided String array (faster if tokens array can be reused)

Tokenize the string into a String array.

Parameters:

   whiteSpace a string containing the whitespace characters

Returns:

   a string array of tokens
tokenize string into a provided String array, SLOW since new array will be constructed

This is the slow-but-convenient Tokenize() method. Slow since the returned string array will be constructed anew with every method call. Consider the Tokenize() method which takes a string array as input, since this may allow reusing of the array, reducing heap allocations.

```cpp
SizeT
Util::String::Tokenize
(const
String
whiteSpace,
&
char
fence,
Array<String>
outTokens
>
&
) const
```

tokenize string, keep strings within fence characters intact (faster if tokens array can be reused)

Tokenize a string, but keeps the string within the fence-character intact. For instance for the sentence:

He said: "I don't know."

A Tokenize(" ", ",", tokens) would return:

token 0: He token 1: said: token 2: I don't know.

```cpp
Array<String>
Util::String::Tokenize
(const
String
whiteSpace,
&
char
fence
) const
```

tokenize string, keep strings within fence characters intact, SLOW since new array will be constructed

Slow version of Tokenize() with fence character. See above Tokenize() for details.

```cpp
String
( IndexT
from,
```
Util::String::ExtractRange
    SizeT   numChars
              const

evoke substring
Excerpt a substring range.

String
Util::String::ExtractToEnd ( IndexT fromIndex ) const

evoke substring to end of this string
Excerpt a substring until the end of the original string.

void
Util::String::Strip (  String charSet )

terminate string at first occurrence of character in set
Terminates the string at the first occurrence of one of the characters in charSet.

IndexT
Util::String::FindStringIndex (  String s,  
                              IndexT startIndex = 0
                           )                                   const

return start index of substring, or InvalidIndex if not found
Return the index of a substring, or InvalidIndex if not found.

IndexT
Util::String::FindCharIndex (  char c,  
                               IndexT startIndex = 0
                            )                                   const

return index of character in string, or InvalidIndex if not found
Return index of character in string, or InvalidIndex if not found.

```cpp
void
Util::String::TerminateAtIndex ( IndexT index )
```

terminate string at given index

Terminates the string at the given index.

```cpp
bool
Util::String::ContainsCharFromSet ( const String charSet ) const
```

returns true if string contains any character from set

Returns true if string contains one of the characters from charset.

```cpp
void
Util::String::TrimLeft ( const String charSet )
```

delete characters from charset at left side of string

Removes all characters in charSet from the left side of the string.

```cpp
void
Util::String::TrimRight ( const String charSet )
```

delete characters from charset at right side of string

Removes all characters in charSet from the right side of the string.

```cpp
void
Util::String::Trim ( const String charSet )
```

trim characters from charset at both sides of string

Trim both sides of a string.

```cpp
void
Util::String::SubstituteString ( const String matchStr,
```
const
String substStr
&
)

substitute every occurrence of a string with another string
Substitute every occurrence of origStr with substStr.

void
Util::String::SubstituteChar (char c,
    char subst
)      [inline]

substitute every occurrence of a character with another character
Replace character with another.

bool
Util::String::CheckValidCharSet (const String charSet ) const [inline]

return true if string only contains characters from charSet argument
Return true if the string only contains characters which are in the defined character set.

bool
Util::String::MatchPattern (const String string,
    const String pattern
)      [static]

pattern matching
Pattern-matching, TCL-style.

IndexT
Util::String::HashCode ( ) const [inline]

return a 32-bit hash code for the string
This method computes a hash code for the string. The method is compatible with the `Util::HashTable` class.

```c
void Util::String::Set (char * str,
     SizeT length)
```

set as char ptr, with explicit length

Sets a new string content. This will handle all special cases and try to minimize heap allocations as much as possible.

```c
int Util::String::AsInt ( ) const
```

return contents as integer

Returns content as integer. Note: this method doesn't check whether the contents is actually a valid integer. Use the IsValidInteger() method for this!

```c
float Util::String::AsFloat ( ) const
```

return contents as float

Returns content as float. Note: this method doesn't check whether the contents is actually a valid float. Use the IsValidFloat() method for this!

```c
Math::float2 Util::String::AsFloat2 ( ) const [inline]
```

return contents as float2

Returns content as float2. Note: this method doesn't check whether the contents is actually a valid float4. Use the IsValidFloat2() method for this!

```c
Math::float4 Util::String::AsFloat4 ( ) const [inline]
```
return contents as float4

Returns content as float4. Note: this method doesn't check whether the contents is actually a valid float4. Use the IsValidFloat4() method for this!

Math::matrix44
Util::String::AsMatrix44 ( ) const

return contents as matrix44

Returns content as matrix44. Note: this method doesn't check whether the contents is actually a valid matrix44. Use the IsValidMatrix44() method for this!

bool
Util::String::IsValidFloat ( ) const [inline]

return true if the content is a valid float

Note: this method is not 100% correct, it just checks for invalid characters.

bool
Util::String::IsValidFloat2 ( ) const [inline]

return true if the content is a valid float2

Note: this method is not 100% correct, it just checks for invalid characters.

bool
Util::String::IsValidFloat4 ( ) const [inline]

return true if the content is a valid float4

Note: this method is not 100% correct, it just checks for invalid characters.

bool
Util::String::IsValidMatrix44 ( ) const [inline]
return true if content is a valid matrix44

Note: this method is not 100% correct, it just checks for invalid characters.

```cpp
String
Util::String::GetFileExtension( ) const
```

get filename extension without dot

Returns:
string representing the filename extension (maybe empty)

```cpp
void
Util::String::ConvertBackslashes( ) [inline]
```

convert backslashes to slashes

Converts backslashes to slashes.

```cpp
void
Util::String::StripFileExtension( )
```

remove file extension

Remove the file extension.

```cpp
String
Util::String::ExtractFileName( ) const [inline]
```

extract the part after the last directory separator

Return a String object containing the part after the last path separator.

```cpp
String
Util::String::ExtractLastDirName( ) const
```

extract the last directory of the path

Return a String object containing the last directory of the path, i.e. a category.
17-Feb-04 floh fixed a bug when the path ended with a slash

```cpp
String Util::String::ExtractDirName() const
```

extract the part before the last directory separator

Return a `String` object containing the part before the last directory separator.

NOTE: I left my fix in that returns the last slash (or colon), this was necessary to tell if a dirname is a normal directory or an assign.

17-Feb-04 floh fixed a bug when the path ended with a slash

```cpp
String Util::String::ExtractToLastSlash() const [inline]
```

extract path until last slash

Return a path string object which contains of the complete path up to the last slash. Returns an empty string if there is no slash in the path.
Util::StringAtom
Util::StringAtom Class Reference

#include <stringatom.h>
Detailed Description

A StringAtom. See StringAtomTableBase for details about the StringAtom system in N3.

TODO: WARNING/STATISTICS for creation from char* or String and converting back to String!

(C) 2007 Radon Labs GmbH
Public Member Functions

**StringAtom** ()
*default constructor*

**StringAtom** (const **StringAtom** &rhs)
*copy constructor*

**StringAtom** (char *ptr)
*construct from char ptr*

**StringAtom** (const char *ptr)
*construct from char ptr*

**StringAtom** (unsigned char *ptr)
*construct from char ptr*

**StringAtom** (const unsigned char *ptr)
*construct from char ptr*

**StringAtom** (const **String** &str)
*construct from string object*

void **operator=** (const **StringAtom** &rhs)
*assignment*

void **operator=** (const char *ptr)
*assignment from char ptr*

void **operator=** (const **String** &str)
*assignment from string object*

bool **operator==** (const **StringAtom** &rhs) const
*equality operator*

bool **operator!=** (const **StringAtom** &rhs) const
*inequality operator*

bool **operator>** (const **StringAtom** &rhs) const
*greater-than operator*

bool **operator<** (const **StringAtom** &rhs) const
*less-than operator*

bool **operator>=** (const **StringAtom** &rhs) const
*greater-or-equal operator*

bool **operator<=** (const **StringAtom** &rhs) const
*less-or-equal operator*

bool **operator==** (const char *rhs) const
equality with char* (SLOW!)

bool operator!= (const char *rhs) const
inequality with char* (SLOW!)

bool operator== (const String &rhs) const
equality with string object (SLOW!)

bool operator!= (const String &rhs) const
inequality with string object (SLOW!)

void Clear ()
clear content (becomes invalid)

bool IsValid () const
return true if valid (contains a non-empty string)

const char * Value () const
get contained string as char ptr (fast)

String AsString () const
get containted string as string object (SLOW!!!)
Member Function Documentation

```cpp
bool Util::StringAtom::operator==(const char* rhs) const

equality with char* (SLOW!)
Compare with raw string. Careful, slow!

bool Util::StringAtom::operator!=(const char* rhs) const

inequality with char* (SLOW!)
Compare with raw string. Careful, slow!

bool Util::StringAtom::operator==(const String& rhs) const [inline]

equality with string object (SLOW!)
Compare with String object. Careful, slow!

bool Util::StringAtom::operator!=(const String& rhs) const [inline]

inequality with string object (SLOW!)
Compare with String object. Careful, slow!

String Util::StringAtom::AsString() const [inline]

get contained string as string object (SLOW!!!)
SLOW!!!
```
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

**Util::StringAtomTableBase**
Util::StringAtomTableBase Class Reference

#include <stringatomtablebase.h>

Inheritance diagram for Util::StringAtomTableBase:
Detailed Description

This implements the base class for thread-local and global string atom table classes.

In order to reduce thread-synchronization, there are 2 levels of string atom tables in N3. One global string atom table for all threads, and one thread-local string atom table in each thread which acts as a cache for the global table. If a new string atom is created from a string, the thread-local string atom table will be searched first. If the string has already been registered in the thread-local table, the string atom will be setup and no locking at all is necessary. Only if the string is not in the thread local table, the global string atom table will be consulted (which requires locking). If the string is in the global table, the pointer to the string will be sorted into the thread-local atom table and the string will be setup. If the string is completely new (not even in the global atom table), then the string needs to be sorted both into the global, and the thread-local atom table.

(C) 2009 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StringAtomTableBase()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~StringAtomTableBase()</code></td>
<td>destructor</td>
</tr>
</tbody>
</table>
Protected Member Functions

const char * Find (const char *str) const

find a string pointer in the atom table
Data Structures

struct **StaticString**

*a static string class for sorting the array* More...
Util::StringAtomTableBase::StaticString
Util::StringAtomTableBase::StaticString

Struct Reference

#include <stringatomtablebase.h>
Detailed Description

a static string class for sorting the array
### Public Member Functions

| bool operator==(const StaticString &rhs) const | equality operator |
| bool operator!=(const StaticString &rhs) const | inequality operator |
| bool operator<(const StaticString &rhs) const | less-than operator |
| bool operator>(const StaticString &rhs) const | greater-than operator |
Util::StringBuffer
Util::StringBuffer Class Reference

#include <stringbuffer.h>
Detailed Description

Global string buffer for the StringAtom system. This is where all raw strings for the StringAtom system are stored. If enabled, the StringBuffer can grow, but it may never shrink. Once a string is in the string buffer, it cannot be removed. String data is simply appended to the last position, strings are separated by a 0-terminator-byte. A string is guaranteed never to move in memory. Several threads can have simultaneous read-access to the string buffer, even while an AddString() is in progress by another thread. Only if several threads attempt to call AddString() a lock must be taken.

NOTE: NOT thread-safe! Usually, GlobalStringAtomTable cares about thread-safety for the global string buffer.

(C) 2009 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StringBuffer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~StringBuffer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <strong>Setup</strong> (SizeT size)</td>
<td>setup the string buffer with size in bytes</td>
</tr>
<tr>
<td>void <strong>Discard</strong> ()</td>
<td>discard the string buffer</td>
</tr>
<tr>
<td>bool <strong>IsValid</strong> () const</td>
<td>return true if string buffer has been setup</td>
</tr>
<tr>
<td>const char * <strong>AddString</strong> (const char *str)</td>
<td>add a string to the end of the string buffer, return pointer to string</td>
</tr>
<tr>
<td>const char * <strong>NextString</strong> (const char *prev)</td>
<td>DEBUG: return next string in string buffer.</td>
</tr>
<tr>
<td>SizeT <strong>GetNumChunks</strong> () const</td>
<td>DEBUG: get number of allocated chunks.</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Util::StringBuffer::Setup ( SizeT size )

setup the string buffer with size in bytes

NOTE: this method must be called before any threads are spawned.
```

```cpp
const char * Util::StringBuffer::AddString ( char * str )

add a string to the end of the string buffer, return pointer to string

Copies a string to the end of the string buffer, returns pointer to copied string.
```
Util::Variant
Util::Variant Class Reference

#include <variant.h>
Detailed Description

An "any type" variable.

Since the Variant class has a rich set of assignment and cast operators, a variant variable can most of the time be used like a normal C++ variable.

(C) 2006 RadonLabs GmbH.
Public Types

enum Type
  variant types
### Public Member Functions

<table>
<thead>
<tr>
<th>Constructor Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Variant()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>Variant(int rhs)</code></td>
<td>int constructor</td>
</tr>
<tr>
<td><code>Variant(float rhs)</code></td>
<td>float constructor</td>
</tr>
<tr>
<td><code>Variant(bool rhs)</code></td>
<td>bool constructor</td>
</tr>
<tr>
<td><code>Variant(const Math::float4 &amp;v)</code></td>
<td>float4 constructor</td>
</tr>
<tr>
<td><code>Variant(const Math::matrix44 &amp;m)</code></td>
<td>matrix44 constructor</td>
</tr>
<tr>
<td><code>Variant(const Util::String &amp;rhs)</code></td>
<td>string constructor</td>
</tr>
<tr>
<td><code>Variant(const Util::Blob &amp;blob)</code></td>
<td>blob constructor</td>
</tr>
<tr>
<td><code>Variant(const Util::Guid &amp;guid)</code></td>
<td>guid constructor</td>
</tr>
<tr>
<td><code>Variant(const char *chrPtr)</code></td>
<td>const char constructor</td>
</tr>
<tr>
<td><code>Variant(Core::RefCounted *ptr)</code></td>
<td>object constructor</td>
</tr>
<tr>
<td><code>Variant(const Util::Array&lt;int&gt; &amp;rhs)</code></td>
<td>int array constructor</td>
</tr>
<tr>
<td><code>Variant(const Util::Array&lt;float&gt; &amp;rhs)</code></td>
<td>float array constructor</td>
</tr>
<tr>
<td><code>Variant(const Util::Array&lt;bool&gt; &amp;rhs)</code></td>
<td>bool array constructor</td>
</tr>
<tr>
<td><code>Variant(const Util::Array&lt;Math::float4&gt; &amp;rhs)</code></td>
<td></td>
</tr>
</tbody>
</table>
### float4 array constructor

**Variant** (const `Util::Array<Math::matrix44>` &rhs)  

### matrix44 array constructor

**Variant** (const `Util::Array<Util::String>` &rhs)  

### string array constructor

**Variant** (const `Util::Array<Util::Blob>` &rhs)  

### blob array constructor

**Variant** (const `Util::Array<Util::Guid>` &rhs)  

### guid array constructor

**Variant** (const **Variant** &rhs)  

### copy constructor

**~Variant** ()  

### destructor

#### void `SetType` (Type t)

set type of attribute

#### Type `GetType` () const

get type

#### void `Clear` ()

clear content, resets type to void

#### void operator= (const **Variant** &rhs)

assignment operator

#### void operator= (int val)

int assignment operator

#### void operator= (float val)

float assignment operator

#### void operator= (bool val)

bool assignment operator

#### void operator= (const `Math::float4` &val)

float4 assignment operator

#### void operator= (const `Math::matrix44` &val)

matrix44 assignment operator

#### void operator= (const `Util::String` &s)
<table>
<thead>
<tr>
<th>Assignment Type</th>
<th>C++ Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>string</strong></td>
<td><code>void operator= (const Util::Blob &amp;val)</code></td>
</tr>
<tr>
<td><strong>blob</strong></td>
<td><code>void operator= (const Util::Blob &amp;val)</code></td>
</tr>
<tr>
<td><strong>guid</strong></td>
<td><code>void operator= (const Util::Guid &amp;val)</code></td>
</tr>
<tr>
<td><strong>char pointer</strong></td>
<td><code>void operator= (const char *chrPtr)</code></td>
</tr>
<tr>
<td><strong>object</strong></td>
<td><code>void operator= (const Core::RefCounted *ptr)</code></td>
</tr>
<tr>
<td><strong>int array</strong></td>
<td><code>void operator= (const Util::Array&lt;int&gt; &amp;rhs)</code></td>
</tr>
<tr>
<td><strong>float array</strong></td>
<td><code>void operator= (const Util::Array&lt;float&gt; &amp;rhs)</code></td>
</tr>
<tr>
<td><strong>bool array</strong></td>
<td><code>void operator= (const Util::Array&lt;bool&gt; &amp;rhs)</code></td>
</tr>
<tr>
<td><strong>float4 array</strong></td>
<td><code>void operator= (const Util::Array&lt;Math::float4&gt; &amp;rhs)</code></td>
</tr>
<tr>
<td><strong>matrix44 array</strong></td>
<td><code>void operator= (const Util::Array&lt;Math::matrix44&gt; &amp;rhs)</code></td>
</tr>
<tr>
<td><strong>string array</strong></td>
<td><code>void operator= (const Util::Array&lt;Util::String&gt; &amp;rhs)</code></td>
</tr>
<tr>
<td><strong>blob array</strong></td>
<td><code>void operator= (const Util::Array&lt;Util::Blob&gt; &amp;rhs)</code></td>
</tr>
<tr>
<td><strong>guid array</strong></td>
<td><code>void operator= (const Util::Array&lt;Util::Guid&gt; &amp;rhs)</code></td>
</tr>
<tr>
<td><strong>bool equality</strong></td>
<td><code>bool operator== (const Variant &amp;rhs)</code></td>
</tr>
<tr>
<td><strong>const equality</strong></td>
<td><code>bool</code></td>
</tr>
<tr>
<td>Operator Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>operator==(int rhs) const</code></td>
<td>int equality operator</td>
</tr>
<tr>
<td><code>operator==(float rhs) const</code></td>
<td>float equality operator</td>
</tr>
<tr>
<td><code>operator==(bool rhs) const</code></td>
<td>bool equality operator</td>
</tr>
<tr>
<td><code>operator==(const Math::float4 &amp;rhs) const</code></td>
<td>float4 equality operator</td>
</tr>
<tr>
<td><code>operator==(const Util::String &amp;rhs) const</code></td>
<td>string equality operator</td>
</tr>
<tr>
<td><code>operator==(const Util::Guid &amp;rhs) const</code></td>
<td>guid equality operator</td>
</tr>
<tr>
<td><code>operator==(const char *chrPtr) const</code></td>
<td>char ptr equality operator</td>
</tr>
<tr>
<td><code>operator==(Core::RefCounted *ptr) const</code></td>
<td>pointer equality operator</td>
</tr>
<tr>
<td><code>operator!=(const Variant &amp;rhs) const</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>operator!=(int rhs) const</code></td>
<td>int inequality operator</td>
</tr>
<tr>
<td><code>operator!=(float rhs) const</code></td>
<td>float inequality operator</td>
</tr>
<tr>
<td><code>operator!=(bool rhs) const</code></td>
<td>bool inequality operator</td>
</tr>
<tr>
<td><code>operator!=(const Math::float4 &amp;rhs) const</code></td>
<td>float4 inequality operator</td>
</tr>
<tr>
<td><code>operator!=(const Util::String &amp;rhs) const</code></td>
<td>string inequality operator</td>
</tr>
</tbody>
</table>
| `operator!=(const Util::Guid &rhs) const` | }
bool operator!= (const char *chrPtr) const

cchar ptr inequality operator

bool operator!= (Core::RefCounted *ptr) const

pointer equality operator

bool operator> (const Variant &rhs) const

greater operator

bool operator< (const Variant &rhs) const

less operator

bool operator>=(const Variant &rhs) const

greater equal operator

bool operator<=(const Variant &rhs) const

less equal operator

void SetInt (int val)

set integer content

int GetInt () const

get integer content

void SetFloat (float val)

set float content

float GetFloat () const

get float content

void SetBool (bool val)

set bool content

bool GetBool () const

get bool content

void SetString (const Util::String &val)

set string content

const Util::String & GetString () const

get string content

void SetFloat4 (const Math::float4 &val)

set float4 content

Math::float4 GetFloat4 () const

get float4 content
<table>
<thead>
<tr>
<th>void SetMatrix44 (const Math::matrix44 &amp;val)</th>
<th>set matrix44 content</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Math::matrix44 &amp; GetMatrix44 () const</td>
<td>get matrix44 content</td>
</tr>
<tr>
<td>void SetBlob (const Util::Blob &amp;val)</td>
<td>set blob</td>
</tr>
<tr>
<td>const Util::Blob &amp; GetBlob () const</td>
<td>get blob</td>
</tr>
<tr>
<td>void SetGuid (const Util::Guid &amp;val)</td>
<td>set guid content</td>
</tr>
<tr>
<td>const Util::Guid &amp; GetGuid () const</td>
<td>get guid content</td>
</tr>
<tr>
<td>void SetObject (Core::RefCounted *ptr)</td>
<td>set object pointer</td>
</tr>
<tr>
<td>Core::RefCounted * GetObject () const</td>
<td>get object pointer</td>
</tr>
<tr>
<td>void SetIntArray (const Util::Array&lt;int&gt; &amp;val)</td>
<td>set int array content</td>
</tr>
<tr>
<td>const Util::Array&lt;int&gt; &amp; GetIntArray () const</td>
<td>get int array content</td>
</tr>
<tr>
<td>void SetFloatArray (const Util::Array&lt;float&gt; &amp;val)</td>
<td>set float array content</td>
</tr>
<tr>
<td>const Util::Array&lt;float&gt; &amp; GetFloatArray () const</td>
<td>get float array content</td>
</tr>
<tr>
<td>void SetBoolArray (const Util::Array&lt;bool&gt; &amp;val)</td>
<td>set bool array content</td>
</tr>
<tr>
<td>const Util::Array&lt;bool&gt; &amp; GetBoolArray () const</td>
<td>get bool array content</td>
</tr>
<tr>
<td>void SetFloat4Array (const Util::Array<a href="">Math::float4</a> &amp;val)</td>
<td>set float4 array content</td>
</tr>
<tr>
<td>const Util::Array<a href="">Math::float4</a> &amp; GetFloat4Array () const</td>
<td>get float4 array content</td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td><strong>void</strong> SetMatrix44Array (const Util::Array&lt; Math::matrix44 &gt; &amp;val)</td>
<td>set matrix44 array content</td>
</tr>
<tr>
<td><strong>const Util::Array&lt; Math::matrix44 &gt; &amp;</strong> GetMatrix44Array () const</td>
<td>get matrix44 array content</td>
</tr>
<tr>
<td><strong>void</strong> SetStringArray (const Util::Array&lt; Util::String &gt; &amp;val)</td>
<td>set string array content</td>
</tr>
<tr>
<td><strong>const Util::Array&lt; Util::String &gt; &amp;</strong> GetStringArray () const</td>
<td>get string array content</td>
</tr>
<tr>
<td><strong>void</strong> SetGuidArray (const Util::Array&lt; Util::Guid &gt; &amp;val)</td>
<td>set guid array content</td>
</tr>
<tr>
<td><strong>const Util::Array&lt; Util::Guid &gt; &amp;</strong> GetGuidArray () const</td>
<td>get guid array content</td>
</tr>
<tr>
<td><strong>void</strong> SetBlobArray (const Util::Array&lt; Util::Blob &gt; &amp;val)</td>
<td>set blob array content</td>
</tr>
<tr>
<td><strong>const Util::Array&lt; Util::Blob &gt; &amp;</strong> GetBlobArray () const</td>
<td>get blob array content</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static Util::String TypeToString (Type t)</code></td>
<td>convert type to string</td>
</tr>
<tr>
<td><code>static Type StringToType (const Util::String &amp;str)</code></td>
<td>convert string to type</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
visibility::observercontext
Visibility::ObserverContext Class Reference

#include <observercontext.h>

Inheritance diagram for Visibility::ObserverContext:

```
   Core::RefCounted
   |     |
   v     v
Visibility::ObserverContext
```
Detailed Description

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ObserverContext</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~ObserverContext ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void Setup (const Ptr&lt; InternalGraphics::InternalGraphicsEntity &gt;&amp; entity)</td>
<td>setup from graphics entity</td>
</tr>
<tr>
<td>Math::ClipStatus::Type</td>
<td><strong>ComputeClipStatus</strong> (const &amp;boundingBox)</td>
</tr>
<tr>
<td>const Ptr&lt; InternalGraphics::InternalGraphicsEntity &gt;&amp;</td>
<td><strong>GetObserverEntity</strong> () const</td>
</tr>
<tr>
<td>Math::matrix44 &amp;</td>
<td><strong>GetProjectionMatrix</strong> ()</td>
</tr>
<tr>
<td>Math::bbox &amp;</td>
<td><strong>GetBoundingBox</strong> ()</td>
</tr>
<tr>
<td>Math::point &amp;</td>
<td><strong>GetPosition</strong> ()</td>
</tr>
<tr>
<td>ObserverCullingType</td>
<td><strong>GetType</strong> () const</td>
</tr>
<tr>
<td>ObserverCullingType *</td>
<td><strong>GetTypeRef</strong> ()</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp; rtti)</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp; className)</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void DumpRefCountingLeaks()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Visibility::VisibilityBoxSystem
Visibility::VisibilityBoxSystem Class Reference

#include <visibilityboxsystem.h>

Inheritance diagram for Visibility::VisibilityBoxSystem:
Detailed Description

The **VisibilityBoxSystem** allows to group entities in visibility boxes. If the camera is inside any box, only this box and all its neighbors (the overlapping boxes) are visible, and therefore its containing entities. If the camera is outside any box, all boxes are visible. The entities are automatically sorted into the surrounded boxes.

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>VisibilityBoxSystem ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~VisibilityBoxSystem ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Open (IndexT orderIndex)</code></td>
<td>Open the graphics server</td>
</tr>
<tr>
<td><code>Close ()</code></td>
<td>Close the graphics server</td>
</tr>
<tr>
<td><code>InsertVisibilityContext (const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</code></td>
<td>Insert entity visibility</td>
</tr>
<tr>
<td><code>RemoveVisibilityContext (const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</code></td>
<td>Remove entity visibility</td>
</tr>
<tr>
<td><code>UpdateVisibilityContext (const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</code></td>
<td>Update entity visibility</td>
</tr>
<tr>
<td><code>InsertVisibilityContainer (const Ptr&lt;VisibilityContainer&gt; &amp;container)</code></td>
<td>Insert visibility container, bunch of contexts with bunch infos</td>
</tr>
<tr>
<td><code>EndAttachVisibilityContainer ()</code></td>
<td>End attach visibility container</td>
</tr>
<tr>
<td><code>CreateVisibilityJob (IndexT frameId, const Ptr&lt;ObserverContext&gt; &amp;observer, Util::FixedArray&lt; Ptr&lt;VisibilityContext&gt; &gt; &gt; &amp;outEntityArray, uint &amp;entityMask)</code></td>
<td>Attach visibility job to port</td>
</tr>
<tr>
<td><code>OnRenderDebug ()</code></td>
<td>Render debug visualizations</td>
</tr>
<tr>
<td><code>GetObserverBitMask ()</code> const</td>
<td>Get observer type mask</td>
</tr>
<tr>
<td><code>isOpen () const</code></td>
<td>Return true if graphics server is open</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>virtual void BeginAttachVisibilityContainer ()</code></td>
<td>Begin attach visibility container</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
Ptr< Jobs::Job > Visibility::VisibilityBoxSystem::CreateVisibilityJob(IndexT frameld,
const Ptr< ObserverContext > observer,
& Util::FixedArray< Ptr< VisibilityContext > > outEntityArray,
& uint & entityMask ) [virtual]
```

attach visibility job to port

Every generated buffer in this function has to doublebuffered!

Reimplemented from Visibility::VisibilitySystemBase.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
```
get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Visibility::VisibilityChecker
Visibility::VisibilityChecker Class Reference

#include <visibilitychecker.h>
Detailed Description

The **VisibilityChecker** manages all visibility systems. Its the interface for all access to the visibility system. If any entity is attached to a stage it will be registered by the checker and added to all attached visibility systems.

If the stage wants to check the visibility it just calls PerformVisibilityQuery which starts a new visibility check with the given observer entity and applies the result of the last frame check.

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VisibilityChecker ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~VisibilityChecker ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Open ()</strong></td>
<td>open</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close</td>
</tr>
<tr>
<td>**void AttachVisibilitySystem (const <strong>&lt;br&gt;</strong>&lt;br&gt;VisibilitySystemBase <strong>&gt;&amp;system)</strong></td>
<td>attach visibility system</td>
</tr>
<tr>
<td><strong>void AttachVisibilitySystems (const <strong>&lt;br&gt;</strong>&lt;br&gt;Util::Array&lt; <strong>&lt;br&gt;</strong>&lt;br&gt;</strong>&lt;br&gt;VisibilitySystemBase <strong>&gt;&amp; &amp;systems)</strong></td>
<td>attach visibility system as array</td>
</tr>
<tr>
<td><strong>void RemoveVisibilitySystem (const <strong>&lt;br&gt;</strong>&lt;br&gt;</strong>&lt;br&gt;**&lt;br&gt;VisibilitySystemBase <strong>&gt;&amp; &amp;system)</strong></td>
<td>remove visibility system</td>
</tr>
<tr>
<td><strong>void RegisterEntity (const <strong>&lt;br&gt;</strong>&lt;br&gt;</strong>&lt;br&gt;InternalGraphics::InternalGraphicsEntity <strong>&gt;&amp; &amp;entity)</strong></td>
<td>register entity</td>
</tr>
<tr>
<td><strong>void UnregisterEntity (const <strong>&lt;br&gt;</strong>&lt;br&gt;</strong>&lt;br&gt;**&lt;br&gt;InternalGraphics::InternalGraphicsEntity <strong>&gt;&amp; &amp;entity)</strong></td>
<td>unregister entity</td>
</tr>
<tr>
<td><strong>void BeginAttachVisibilityContainer ()</strong></td>
<td>begin attach visibility container</td>
</tr>
<tr>
<td><strong>void AttachVisibilityContainer (const <strong>&lt;br&gt;</strong>&lt;br&gt;</strong>&lt;br&gt;**&lt;br&gt;VisibilityContainer <strong>&gt;&amp; &amp;container)</strong></td>
<td>attach visibility container, alternative method for registering a bunch of entities</td>
</tr>
<tr>
<td><strong>void EndAttachVisibilityContainer ()</strong></td>
<td>end attach visibility container</td>
</tr>
<tr>
<td><strong>void UpdateVisibilityContext (const <strong>&lt;br&gt;</strong>&lt;br&gt;</strong>&lt;br&gt;<strong>&lt;br&gt;</strong>&lt;br&gt;**&lt;br&gt;InternalGraphics::InternalGraphicsEntity <strong>&gt;&amp; &amp;entity)</strong></td>
<td>update entity visibility context on a transform change</td>
</tr>
<tr>
<td>function</td>
<td>Signature</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>void PerformVisibilityQuery</td>
<td>IndexT frameId, const Ptr<a href="">InternalGraphics::InternalGraphicsEntity</a> &amp;observerEntity, uint entityMask</td>
</tr>
<tr>
<td>void ClearVisibilityLinks</td>
<td>(InternalGraphics::InternalGraphicsEntity::LinkType linkType)</td>
</tr>
<tr>
<td>void OnRenderDebug</td>
<td></td>
</tr>
</tbody>
</table>
Visibility::VisibilityClusterSystem
Visibility::VisibilityClusterSystem Class Reference

#include <visibilityclustersystem.h>

Inheritance diagram for Visibility::VisibilityClusterSystem:
Detailed Description

Culls the attached graphics entities if the viewer is inside a bounding box cluster. VisibilityClusters are created and manually configured by level designers inside the level editor.

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>VisibilityClusterSystem()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~VisibilityClusterSystem()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual void Open(IndexT orderIndex)</code></td>
<td>Open the graphics server</td>
</tr>
<tr>
<td><code>virtual void Close()</code></td>
<td>Close the graphics server</td>
</tr>
<tr>
<td><code>virtual void InsertVisibilityContext(const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</code></td>
<td>Insert entity visibility</td>
</tr>
<tr>
<td><code>virtual void RemoveVisibilityContext(const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</code></td>
<td>Remove entity visibility</td>
</tr>
<tr>
<td><code>virtual void UpdateVisibilityContext(const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</code></td>
<td>Update entity visibility</td>
</tr>
<tr>
<td><code>virtual void InsertVisibilityContainer(const Ptr&lt;VisibilityContainer&gt; &amp;container)</code></td>
<td>Insert visibility container, bunch of contexts with bunch infos</td>
</tr>
<tr>
<td><code>virtual Ptr&lt;Jobs::Job&gt; CreateVisibilityJob(IndexT frameId, const Ptr&lt;ObserverContext&gt; &amp;observer, Util::FixedArray&lt;Ptr&lt;VisibilityContext&gt;&gt; &amp;outEntityArray, uint &amp;entityMask)</code></td>
<td>Attach visibility job to port</td>
</tr>
<tr>
<td><code>virtual void OnRenderDebug()</code></td>
<td>Render debug visualizations</td>
</tr>
<tr>
<td><code>virtual uint GetObserverBitMask()</code></td>
<td>Get observer type mask</td>
</tr>
<tr>
<td><code>bool IsOpen()</code></td>
<td>Return true if graphics server is open</td>
</tr>
<tr>
<td><code>virtual void BeginAttachVisibilityContainer()</code></td>
<td>Begin attach visibility container</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void EndAttachVisibilityContainer()</td>
<td>end attach visibility container</td>
</tr>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName() const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC() const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Visibility::VisibilityContainer
Visibility::VisibilityContainer Class Reference

#include <visibilitycontainer.h>

Inheritance diagram for Visibility::VisibilityContainer:
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>VisibilityContainer ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~VisibilityContainer ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>SetGraphicsEntities(const Util::Array&lt;Ptr&lt;InternalGraphics::InternalGraphicsEntity &gt;&gt; &amp;entities)</code></td>
<td>Set gfx entities</td>
</tr>
<tr>
<td>const <code>Util::Array&lt;Ptr&lt;VisibilityContext&gt; &gt; &amp; GetVisibilityContexts () const</code></td>
<td>Get visibility contexts</td>
</tr>
<tr>
<td><code>GetRefCount () const</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release ()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>Return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class, by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, by string, derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, by fourcc, derived class</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const Util::Array&lt; Ptr&lt; InternalGraphics::InternalGraphicsEntity &gt; &gt; &amp; GetGraphicsEntities() const</code></td>
<td>get entities</td>
</tr>
<tr>
<td><code>void SetVisibilityContexts(const Util::Array&lt; Ptr&lt; VisibilityContext &gt; &amp;contexts)</code></td>
<td>convert gfx entities to contexts</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
```
builds only!

This method should be called as the very last before an application exits.
Visibility::VisibilityContext
Visibility::VisibilityContext Class Reference

#include <visibilitycontext.h>

Inheritance diagram for Visibility::VisibilityContext:
Detailed Description

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>VisibilityContext()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~VisibilityContext()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>const Ptr&lt; InternalGraphics::InternalGraphicsEntity &gt; &amp; GetGfxEntity()</code></td>
<td>Get GfxEntity</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetBoundingBox()</code></td>
<td>get BoundingBox</td>
</tr>
<tr>
<td><code>const IndexT GetVisibleFrameId()</code></td>
<td>get frameId visible</td>
</tr>
<tr>
<td><code>void SetVisibleFrameId(IndexT frameId)</code></td>
<td>set Visible</td>
</tr>
<tr>
<td><code>int GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf(const Util::FourCC)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class or a derived class by <code>Rtti</code></td>
</tr>
<tr>
<td><code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>Get the class <code>FourCC</code> code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Visibility::VisibilityQuadtree
Visibility::VisibilityQuadtree Class Reference

#include <visibilityquadtree.h>

Inheritance diagram for Visibility::VisibilityQuadtree:
Detailed Description

Simple quadtree for culling. Entities are automatically sorted into quadtree.

(C) 2010 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>VisibilityQuadtree()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~VisibilityQuadtree()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>SetQuadTreeSettings(uchar depth, const Math::bbox &amp;worldBBox)</code></td>
<td>set quad tree depth and bounding box</td>
</tr>
<tr>
<td>virtual void <code>Open(IndexT orderIndex)</code></td>
<td>open the graphics server</td>
</tr>
<tr>
<td>virtual void <code>Close()</code></td>
<td>close the graphics server</td>
</tr>
<tr>
<td>virtual void <code>InsertVisibilityContext(const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</code></td>
<td>insert entity visibility</td>
</tr>
<tr>
<td>virtual void <code>RemoveVisibilityContext(const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</code></td>
<td>remove entity visibility</td>
</tr>
<tr>
<td>virtual void <code>UpdateVisibilityContext(const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</code></td>
<td>update entity visibility</td>
</tr>
<tr>
<td>virtual <code>Ptr&lt;Jobs::Job&gt;</code> <code>CreateVisibilityJob(IndexT frameId, const Ptr&lt;ObserverContext&gt; &amp;observer, Util::FixedArray&lt;Ptr&lt;VisibilityContext&gt; &gt; &amp;outEntitiyArray, uint &amp;entityMask)</code></td>
<td>attach visibility job to port</td>
</tr>
<tr>
<td>virtual void <code>OnRenderDebug()</code></td>
<td>render debug visualizations</td>
</tr>
<tr>
<td>virtual uint <code>GetObserverBitMask()</code> const</td>
<td>get observer type mask</td>
</tr>
<tr>
<td>bool <code>IsOpen()</code> const</td>
<td>return true if graphics server is open</td>
</tr>
<tr>
<td>virtual void <code>BeginAttachVisibilityContainer()</code></td>
<td>begin attach visibility container</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void InsertVisibilityContainer</td>
<td>attach visibility container, alternative method for registering a bunch of entities</td>
</tr>
<tr>
<td>virtual void EndAttachVisibilityContainer()</td>
<td>end attach visibility container</td>
</tr>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf(const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName() const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC() const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Visibility::VisibilityQuery
Visibility::VisibilityQuery Class Reference

#include <visibilityquery.h>

Inheritance diagram for Visibility::VisibilityQuery:
Detailed Description

Each VisibilityQuery starts dependend jobs for every attached visibility system.

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VisibilityQuery ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~VisibilityQuery ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>AttachVisibilitySystem</strong> (const VisibilitySystemBase &amp;visSystem) &amp;</td>
<td>Attach visible system, used by this job</td>
</tr>
<tr>
<td><strong>SetObserver</strong> (const Ptr&lt; InternalGraphics::InternalGraphicsEntity &gt; &amp;val) &amp;</td>
<td>Set Observer</td>
</tr>
<tr>
<td>const <strong>GetObserver</strong> () const</td>
<td>Get Observer</td>
</tr>
<tr>
<td><strong>Run</strong> (IndexT frameId) &amp;</td>
<td>Run job</td>
</tr>
<tr>
<td><strong>IsFinished</strong> () const &amp;</td>
<td>Is finished</td>
</tr>
<tr>
<td><strong>WaitForFinished</strong> () const &amp;</td>
<td>Wait for finished</td>
</tr>
<tr>
<td>InternalGraphics::InternalGraphicsEntity::LinkType <strong>GetObserverType</strong> ()</td>
<td>Get observer link type</td>
</tr>
<tr>
<td>const <strong>GetVisibleEntities</strong> () const &amp;</td>
<td>Get visible entities</td>
</tr>
<tr>
<td><strong>SetEntityMask</strong> (uint mask) &amp;</td>
<td>Set entity mask</td>
</tr>
<tr>
<td><strong>GetRefCount</strong> () const</td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><strong>AddRef</strong> () &amp;</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><strong>Release</strong> () &amp;</td>
<td>Decrease refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;)</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rtti) const</code></td>
<td>return true if this object is instance of given derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rtti) const</code></td>
<td>return true if this object is instance of given derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
```
builds only!

This method should be called as the very last before an application exits.
Visibility::VisibilitySystemBase
Visibility::VisibilitySystemBase Class Reference

#include <visibilitysystembase.h>

Inheritance diagram for Visibility::VisibilitySystemBase:
Detailed Description

(C) 2010 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VisibilitySystemBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~VisibilitySystemBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual void Open (IndexT orderIndex)</strong></td>
<td>open the graphics server</td>
</tr>
<tr>
<td><strong>virtual void Close ()</strong></td>
<td>close the graphics server</td>
</tr>
<tr>
<td><strong>bool IsOpen () const</strong></td>
<td>return true if graphics server is open</td>
</tr>
<tr>
<td><strong>virtual void InsertVisibilityContext (const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</strong></td>
<td>insert entity visibility</td>
</tr>
<tr>
<td><strong>virtual void RemoveVisibilityContext (const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</strong></td>
<td>remove entity visibility</td>
</tr>
<tr>
<td><strong>virtual void UpdateVisibilityContext (const Ptr&lt;VisibilityContext&gt; &amp;entityVis)</strong></td>
<td>update entity visibility</td>
</tr>
<tr>
<td><strong>virtual void BeginAttachVisibilityContainer ()</strong></td>
<td>begin attach visibility container</td>
</tr>
<tr>
<td><strong>virtual void InsertVisibilityContainer (const Ptr&lt;VisibilityContainer&gt; &amp;container)</strong></td>
<td>attach visibility container, alternative method for registering a bunch of entities</td>
</tr>
<tr>
<td><strong>virtual void EndAttachVisibilityContainer ()</strong></td>
<td>end attach visibility container</td>
</tr>
<tr>
<td><strong>virtual Ptr&lt; Jobs::Job &gt;</strong> CreateVisibilityJob (IndexT frameId, const Ptr&lt;ObserverContext&gt; &amp;observer, Util::FixedArray&lt; Ptr&lt;VisibilityContext&gt; &gt; &gt; &amp;outEntityArray, uint &amp;entityMask)**</td>
<td>attach visibility job to port</td>
</tr>
<tr>
<td><strong>virtual void OnRenderDebug ()</strong></td>
<td>render debug visualizations</td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual uint <code>GetObserverBitMask</code>() const</td>
<td>get observer type mask</td>
</tr>
<tr>
<td>int <code>GetRefCount</code>() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef</code>()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <code>Release</code>()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Util::String</code> &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <code>IsInstanceOf</code> (const <code>Util::FourCC</code> &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Util::String</code> &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp; <code>GetClassName</code>() const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC</code> <code>GetClassFourCC</code>() const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
WeakPtr< TYPE > Class Template Reference

#include <weakptr.h>
Detailed Description

template<class TYPE>
class WeakPtr< TYPE >

A smart pointer which does not change the reference count of the target object. Use this to prevent cyclic dependencies between objects. NOTE: The weak ptr only has a subset of methods of Ptr<>. 

(C) 2008 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>WeakPtr ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>WeakPtr (TYPE *p)</code></td>
<td>construct from C++ pointer</td>
</tr>
<tr>
<td><code>WeakPtr (const Ptr&lt; TYPE &gt; &amp;p)</code></td>
<td>construct from Ptr&lt;&gt; pointer</td>
</tr>
<tr>
<td><code>WeakPtr (const WeakPtr&lt; TYPE &gt; &amp;p)</code></td>
<td>construct from WeakPtr&lt;&gt; pointer</td>
</tr>
<tr>
<td><code>~WeakPtr ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void operator= (const Ptr&lt; TYPE &gt; &amp;rhs)</code></td>
<td>assignment operator from Ptr&lt;&gt;</td>
</tr>
<tr>
<td><code>void operator= (const WeakPtr&lt; TYPE &gt; &amp;rhs)</code></td>
<td>assignment operator from WeakPtr&lt;&gt;</td>
</tr>
<tr>
<td><code>void operator= (TYPE *rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td><code>TYPE * operator-&gt; () const</code></td>
<td>safe -&gt; operator</td>
</tr>
<tr>
<td><code>TYPE &amp; operator * () const</code></td>
<td>safe dereference operator</td>
</tr>
<tr>
<td><code>operator TYPE * () const</code></td>
<td>safe pointer cast operator</td>
</tr>
<tr>
<td><code>bool isValid () const</code></td>
<td>check if pointer is valid</td>
</tr>
<tr>
<td><code>TYPE * get () const</code></td>
<td>return direct pointer (asserts if null pointer)</td>
</tr>
<tr>
<td><code>TYPE * get_unsafe () const</code></td>
<td>return direct pointer (returns null pointer)</td>
</tr>
</tbody>
</table>
Win32::SysFunc
Win32::SysFunc Class Reference

#include <win32sysfunc.h>
Detailed Description

Provides **Win32** specific helper functions.

(C) 2006 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Setup()</code></td>
<td>setup low level static objects (must be called before spawning any threads)</td>
</tr>
<tr>
<td><code>Exit(int exitCode)</code></td>
<td>exit process, and to proper cleanup, memleak reporting, etc...</td>
</tr>
<tr>
<td><code>Error(const char *error)</code></td>
<td>display an error message box</td>
</tr>
<tr>
<td><code>MessageBox(const char *msg)</code></td>
<td>display a message box which needs to be confirmed by the user</td>
</tr>
<tr>
<td><code>DebugOut(const char *msg)</code></td>
<td>print a message on the debug console</td>
</tr>
<tr>
<td><code>Sleep(double sec)</code></td>
<td>sleep for a specified amount of seconds</td>
</tr>
<tr>
<td><code>GetSystemInfo()</code></td>
<td>get system info</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:50 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win32::Win32ConsoleHandler
Win32::Win32ConsoleHandler Class Reference

#include <win32consolehandler.h>

Inheritance diagram for Win32::Win32ConsoleHandler:
Detailed Description

The default console handler for Win32, puts normal messages to the debug output channel, and error messages into a message box. Does not provide any input.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Win32ConsoleHandler ()</strong></td>
<td><strong>constructor</strong></td>
</tr>
<tr>
<td>virtual void <strong>Print</strong> (const <strong>Util::String</strong> &amp;s)</td>
<td>called by console to output data</td>
</tr>
<tr>
<td>virtual void <strong>Error</strong> (const <strong>Util::String</strong> &amp;s)</td>
<td>called by console with serious error</td>
</tr>
<tr>
<td>virtual void <strong>Warning</strong> (const <strong>Util::String</strong> &amp;s)</td>
<td>called by console to output warning</td>
</tr>
<tr>
<td>virtual void <strong>Confirm</strong> (const <strong>Util::String</strong> &amp;s)</td>
<td>called by console to display confirmation message box</td>
</tr>
<tr>
<td>virtual void <strong>DebugOut</strong> (const <strong>Util::String</strong> &amp;s)</td>
<td>called by console to output debug string</td>
</tr>
<tr>
<td>virtual bool <strong>HasInput</strong> ()</td>
<td>return true if input is available</td>
</tr>
<tr>
<td>virtual <strong>Util::String</strong> <strong>GetInput</strong> ()</td>
<td>read available input</td>
</tr>
<tr>
<td>virtual void <strong>Open</strong> ()</td>
<td>called by console when attached</td>
</tr>
<tr>
<td>virtual void <strong>Close</strong> ()</td>
<td>called by console when removed</td>
</tr>
<tr>
<td>bool <strong>IsOpen</strong> () const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td>virtual void <strong>Update</strong> ()</td>
<td>called by <strong>Console::Update()</strong></td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong> (const <strong>Rtti</strong> &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong> (const <strong>Util::String</strong> &amp;className)</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

bool
Win32::Win32ConsoleHandler::HasInput ( ) [virtual]

return true if input is available

Since we are blocking the app waiting for user input, we always provide input.

Reimplemented from IO::ConsoleHandler.

String
Win32::Win32ConsoleHandler::GetInput( ) [virtual]

read available input

Get user input from the console.

Reimplemented from IO::ConsoleHandler.

int
Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]

dump the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]

dump the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Win32::Win32Cpu
Win32::Win32Cpu Class Reference

#include <win32cpu.h>
Detailed Description

CPU related definitions for the Win32 platform.

(C) 2008 Radon Labs GmbH
<table>
<thead>
<tr>
<th>static const CoreId</th>
<th><strong>InvalidCoreId</strong> = 0xffffffff</th>
</tr>
</thead>
</table>

core identifiers, under Win32, we basically don't care...

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Win32::Win32DisplayDevice
Win32::Win32DisplayDevice Class Reference

#include <win32displaydevice.h>

Inheritance diagram for Win32::Win32DisplayDevice:
Detailed Description

Win32 implementation of DisplayDevice class. Manages the application window.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Win32DisplayDevice ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~Win32DisplayDevice ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>virtual bool Open ()</strong></td>
<td>Open the display</td>
</tr>
<tr>
<td><strong>virtual void Close ()</strong></td>
<td>Close the display</td>
</tr>
<tr>
<td><strong>virtual void ProcessWindowMessages ()</strong></td>
<td>Process window system messages, call this method once per frame</td>
</tr>
<tr>
<td><strong>HWND GetHwnd () const</strong></td>
<td>Get the application window HWND</td>
</tr>
<tr>
<td><strong>bool AdapterExists (CoreGraphics::Adapter::Code adapter)</strong></td>
<td>Return true if adapter exists</td>
</tr>
<tr>
<td><strong>Util::Array&lt; CoreGraphics::DisplayMode &gt; GetAvailableDisplayModes (CoreGraphics::Adapter::Code adapter, CoreGraphics::PixelFormat::Code pixelFormat)</strong></td>
<td>Get available display modes on given adapter</td>
</tr>
<tr>
<td><strong>bool SupportsDisplayMode (CoreGraphics::Adapter::Code adapter, const CoreGraphics::DisplayMode &amp;requestedMode)</strong></td>
<td>Return true if a given display mode is supported</td>
</tr>
<tr>
<td><strong>CoreGraphics::DisplayMode GetCurrentAdapterDisplayMode (CoreGraphics::Adapter::Code adapter)</strong></td>
<td>Get current adapter display mode (i.e. the desktop display mode)</td>
</tr>
<tr>
<td><strong>CoreGraphics::AdapterInfo GetAdapterInfo (CoreGraphics::Adapter::Code adapter)</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>SetAdapter(CoreGraphics::Adapter::Code a)</code></td>
<td>Set display adapter (make sure adapter exists!)</td>
</tr>
<tr>
<td><code>GetAdapter()</code></td>
<td>Get display adapter</td>
</tr>
<tr>
<td><code>SetDisplayMode(const CoreGraphics::DisplayMode &amp;m)</code></td>
<td>Set display mode (make sure the display mode is supported!)</td>
</tr>
<tr>
<td><code>GetDisplayMode()</code></td>
<td>Get display mode</td>
</tr>
<tr>
<td><code>SetAntiAliasQuality(CoreGraphics::AntiAliasQuality::Code aa)</code></td>
<td>Set antialias quality</td>
</tr>
<tr>
<td><code>GetAntiAliasQuality()</code></td>
<td>Get antialias quality</td>
</tr>
<tr>
<td><code>SetFullscreen(bool b)</code></td>
<td>Set windowed/fullscreen mode</td>
</tr>
<tr>
<td><code>IsFullscreen()</code></td>
<td>Get windowed/fullscreen mode</td>
</tr>
<tr>
<td><code>SetDisplayModeSwitchEnabled(bool b)</code></td>
<td>Enable display mode switch when running fullscreen (default is true);</td>
</tr>
<tr>
<td><code>IsDisplayModeSwitchEnabled()</code></td>
<td>Is display mode switch enabled for fullscreen?</td>
</tr>
<tr>
<td><code>SetTripleBufferingEnabled(bool b)</code></td>
<td>Enable triple buffer for fullscreen (default is double buffering)</td>
</tr>
<tr>
<td><code>IsTripleBufferingEnabled()</code></td>
<td>Is triple buffer enabled for fullscreen?</td>
</tr>
<tr>
<td><code>SetAlwaysOnTop(bool b)</code></td>
<td>Set always-on-top behaviour</td>
</tr>
<tr>
<td><code>IsAlwaysOnTop()</code></td>
<td>Get always-on-top behaviour</td>
</tr>
<tr>
<td><code>SetVerticalSyncEnabled(bool b)</code></td>
<td>Set vertical sync enabled for fullscreen</td>
</tr>
</tbody>
</table>

*Note: `CoreGraphics::` is a placeholder for the actual namespace.*
bool IsVerticalSyncEnabled () const
get vertical sync flag

void SetIconName (const Util::String &s)
set optional window icon resource name

const Util::String & GetIconName () const
get optional window icon resource name

void SetParentWindow (void *h)
set optional parent window handle

void * GetParentWindow () const
get optional parent window handle

void SetWindowTitle (const Util::String &t)
set window title string (can be changed anytime)

const Util::String & GetWindowTitle () const
get window title string

bool IsOpen () const
return true if display is currently open

void AttachEventHandler (const Ptr<CoreGraphics::DisplayEventHandler &h>)
attach a display event handler

void RemoveEventHandler (const Ptr<CoreGraphics::DisplayEventHandler &h})
remove a display event handler

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, derived class</td>
</tr>
<tr>
<td><code>IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, derived class, by string</td>
</tr>
<tr>
<td><code>IsA (const Util::FourCC &amp;rttiFourCC const)</code></td>
<td>return true if this object is instance of given class, derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool <strong>OpenWindow</strong> ()</td>
<td>open the application window</td>
</tr>
<tr>
<td>virtual void <strong>CloseWindow</strong> ()</td>
<td>close the application window</td>
</tr>
<tr>
<td>virtual void <strong>OnMinimized</strong> ()</td>
<td>called on WM_SIZE when window is minimized</td>
</tr>
<tr>
<td>virtual void <strong>OnRestored</strong> ()</td>
<td>called on WM_SIZE when window is restored</td>
</tr>
<tr>
<td>virtual bool <strong>OnSetCursor</strong> ()</td>
<td>called on WM_SETCURSOR</td>
</tr>
<tr>
<td>virtual void <strong>OnPaint</strong> ()</td>
<td>called on WM_PAINT</td>
</tr>
<tr>
<td>virtual void <strong>OnSetFocus</strong> ()</td>
<td>called on WM_SETFOCUS</td>
</tr>
<tr>
<td>virtual void <strong>OnKillFocus</strong> ()</td>
<td>called on WM_KILLFOCUS</td>
</tr>
<tr>
<td>virtual void <strong>OnCloseRequested</strong> ()</td>
<td>called on WM_CLOSE to request if window should be closed</td>
</tr>
<tr>
<td>virtual void <strong>OnToggleFullscreenWindowed</strong> ()</td>
<td>called when Alt-Enter is pressed</td>
</tr>
<tr>
<td>virtual void <strong>OnKeyDown</strong> (WPARAM wParam)</td>
<td>called on WM_KEYDOWN</td>
</tr>
<tr>
<td>virtual void <strong>OnKeyUp</strong> (WPARAM wParam)</td>
<td>called on WM_KEYUP</td>
</tr>
<tr>
<td>virtual void <strong>OnChar</strong> (WPARAM wParam)</td>
<td>called on WM_CHAR</td>
</tr>
<tr>
<td>virtual void <strong>OnMouseButton</strong> (UINT uMsg, LPARAM lParam)</td>
<td>called on mouse button event</td>
</tr>
<tr>
<td>virtual void <strong>OnMouseMove</strong> (LPARAM lParam)</td>
<td>called on WM_MOUSEMOVE</td>
</tr>
<tr>
<td>virtual void <strong>OnMouseWheel</strong> (WPARAM wParam)</td>
<td>called on WM_MOUSEWHEEL</td>
</tr>
</tbody>
</table>
Input::Key::Code: **TranslateKeyCode** (WPARAM wParam) const

- **Called on WM_MOUSEWHEEL**
- **Translate a Windows virtual key code into a Nebula3 key code**

Virtual CoreGraphics::DisplayMode: **ComputeAdjustedWindowRect** ()

- **Adjust window size taking client area into account**

Math::float2: **ComputeAbsMousePos** (LPARAM lParam) const

- **Compute absolute mouse position from lParam**

Math::float2: **ComputeNormMousePos** (const Math::float2 &absMousePos) const

- **Compute normalized mouse position from absolute mouse pos**

Bool: **NotifyEventHandlers** (const CoreGraphics::DisplayEvent &e)

- **Notify event handlers about an event**
## Static Protected Member Functions

<table>
<thead>
<tr>
<th>static LRESULT CALLBACK</th>
<th>WinProc (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)</th>
</tr>
</thead>
<tbody>
<tr>
<td>the WinProc</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Win32::Win32DisplayDevice::ProcessWindowMessages( ) [virtual]

process window system messages, call this method once per frame

Polls for and processes window messages. Call this message once per frame in your render loop. If the user clicks the window close button, or hits Alt-F4, a CloseRequested input event will be sent.

Reimplemented from Base::DisplayDeviceBase.

LRESULT CALLBACK
Win32::Win32DisplayDevice::WinProc( HWND hWnd,
UINT uMsg,
WPARAM wParam,
LPARAM lParam ) [static, protected]

the WinProc

The Nebula3 WinProc.

bool
Win32::Win32DisplayDevice::OpenWindow( ) [protected, virtual]

open the application window

Open the application window.

void
Win32::Win32DisplayDevice::CloseWindow( ) [protected, virtual]

close the application window

Close the application window.

Input::Key::Code
Win32::Win32DisplayDevice::TranslateKeyCode( WPARAM wParam ) const [protected]
translate a Windows virtual key code into a Nebula3 key code

Helper method which translates a Win32 virtual key code into a Nebula key code.

DisplayMode
Win32::Win32DisplayDevice::ComputeAdjustedWindowRect

adjust window size taking client area into account

This will return an adjusted window size which takes the client area of the window into account. This is only relevant for windowed mode.

bool
Base::DisplayDeviceBase::AdapterExists

return true if adapter exists

Checks if the given adapter exists.

Reimplemented in Direct3D9::D3D9DisplayDevice.

Util::Array<DisplayMode>
Base::DisplayDeviceBase::GetAvailableDisplayModes

get available display modes on given adapter

Returns the display modes on the given adapter in the given pixel format.

Reimplemented in Direct3D9::D3D9DisplayDevice.

bool
Base::DisplayDeviceBase::SupportsDisplayMode

return true if a given display mode is supported
This method checks the available display modes on the given adapter against the requested display modes and returns true if the display mode exists.

Reimplemented in Direct3D9::D3D9DisplayDevice.

**DisplayMode**

Base::DisplayDeviceBase::GetCurrentAdapterDisplayMode (CoreGraphics::Adapter::Code adapter)

get current adapter display mode (i.e. the desktop display mode)

This method returns the current adapter display mode. It can be used to get the current desktop display mode.

Reimplemented in Direct3D9::D3D9DisplayDevice.

**AdapterInfo**

Base::DisplayDeviceBase::GetAdapterInfo (CoreGraphics::Adapter::Code adapter) [inherited]

get general info about display adapter

Returns information about the provided adapter.

Reimplemented in Direct3D9::D3D9DisplayDevice.

**void**

Base::DisplayDeviceBase::SetWindowTitle (const Util::String & str) [inherited]

set window title string (can be changed anytime)

Set the window title. An application should be able to change the window title at any time, that's why this is a virtual method, so that a subclass may override it.

**void**

Base::DisplayDeviceBase::AttachEventHandler (const Ptr< CoreGraphics::DisplayEventHandler > h) [inherited]

attach a display event handler

Attach an event handler to the display device.
void Base::DisplayDeviceBase::RemoveEventHandler(const 
    CoreGraphics::DisplayEventHandler h) [inherited]

remove a display event handler

Remove an event handler from the display device.

bool Base::DisplayDeviceBase::NotifyEventHandlers(const 
    CoreGraphics::DisplayEvent e) [protected, inherited]

notify event handlers about an event

Notify all event handlers about an event.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.
Util::FourCC
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:51 2010
Win32::Win32Guid
Win32::Win32Guid Class Reference

#include <win32guid.h>
Detailed Description

Win32 implementation of the Util::Guid class. GUIDs can be compared and provide a hash code, so they can be used as keys in most collections.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void * operator new (size_t s)</td>
<td>override new operator</td>
</tr>
<tr>
<td>void operator delete (void *ptr)</td>
<td>override delete operator</td>
</tr>
<tr>
<td>Win32Guid ()</td>
<td>constructor</td>
</tr>
<tr>
<td>Win32Guid (const Win32Guid &amp;rhs)</td>
<td>copy constructor</td>
</tr>
<tr>
<td>Win32Guid (const unsigned char *ptr, SizeT size)</td>
<td>construct from raw binary data as returned by AsBinary()</td>
</tr>
<tr>
<td>void operator= (const Win32Guid &amp;rhs)</td>
<td>assignment operator</td>
</tr>
<tr>
<td>void operator= (const Util::String &amp;rhs)</td>
<td>assignment operator from string</td>
</tr>
<tr>
<td>bool operator== (const Win32Guid &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool operator!= (const Win32Guid &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td>bool operator&lt; (const Win32Guid &amp;rhs) const</td>
<td>less-than operator</td>
</tr>
<tr>
<td>bool operator&lt;= (const Win32Guid &amp;rhs) const</td>
<td>less-or-equal operator</td>
</tr>
<tr>
<td>bool operator&gt; (const Win32Guid &amp;rhs) const</td>
<td>greater-than operator</td>
</tr>
<tr>
<td>bool operator&gt;= (const Win32Guid &amp;rhs) const</td>
<td>greater-or-equal operator</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>return true if the contained guid is valid (not NIL)</td>
</tr>
<tr>
<td>void Generate ()</td>
<td>generate a new guid</td>
</tr>
<tr>
<td>Util::String AsString () const</td>
<td>get as string</td>
</tr>
<tr>
<td>SizeT AsBinary (const unsigned char *outPtr) const</td>
<td></td>
</tr>
<tr>
<td>IndexT</td>
<td><strong>HashCode () const</strong></td>
</tr>
<tr>
<td>--------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td><em>get a hash code (compatible with Util::HashTable)</em></td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FromString</code></td>
<td>construct from string representation</td>
</tr>
<tr>
<td><code>FromBinary</code></td>
<td>construct from binary representation</td>
</tr>
<tr>
<td><code>BinarySize</code></td>
<td>return the size of the binary representation in bytes</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Win32Guid**

```cpp
class Win32Guid
{
public:
    static Win32Guid::FromBinary
        ( const unsigned char * ptr,
          SizeT numBytes );
};
```

construct from binary representation

Constructs the guid from binary data, as returned by the `AsBinary()`.

```cpp
SizeT Win32::Win32Guid::AsBinary
    ( const unsigned char *& outPtr ) const
```

get pointer to binary data

This method allows read access to the raw binary data of the uuid. It returns the number of bytes in the buffer, and a pointer to the data.

```cpp
IndexT Win32::Win32Guid::HashCode
    ( ) const
```

get a hash code (compatible with `Util::HashTable`

This method returns a hash code for the uuid, compatible with `Util::HashTable`.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Win32::Win32InputDisplayEventHandler
Win32::Win32InputDisplayEventHandler
Class Reference

#include <win32inputdisplayeventhandler.h>

Inheritance diagram for Win32::Win32InputDisplayEventHandler:
Detailed Description

Translates DisplayEvents that are relevant for the input system into InputEvents.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual bool HandleEvent (const CoreGraphics::DisplayEvent &amp;event)</code></td>
<td>called when an event happens</td>
</tr>
<tr>
<td><code>virtual bool PutEvent (const DisplayEvent &amp;event)</code></td>
<td>called by DisplayDevice when an event happens</td>
</tr>
<tr>
<td><code>void HandlePendingEvents ()</code></td>
<td>handle all pending events (called by consumer thread)</td>
</tr>
<tr>
<td><code>virtual void OnAttach ()</code></td>
<td>called when the event handler is attached to the DisplayDevice</td>
</tr>
<tr>
<td><code>virtual void OnRemove ()</code></td>
<td>called when the event handler is removed from the DisplayDevice</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td></td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC () const</strong></td>
</tr>
<tr>
<td>get the class FourCC code</td>
<td></td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

```cpp
bool CoreGraphics::ThreadSafeDisplayEventHandler::PutEvent (const DisplayEvent & e ) [virtual, inherited]

called by DisplayDevice when an event happens

Put an event into the event handler. This method is called by the render thread's DisplayDevice. Events are queued until the consumer thread processes them by calling HandlePendingEvents().

Reimplemented from CoreGraphics::DisplayEventHandler.

```cpp
void CoreGraphics::ThreadSafeDisplayEventHandler::HandlePendingEvents ( ) [inherited]

handle all pending events (called by consumer thread)

Process pending events. This method should be called frequently by the consumer thread. Pending events will be dequeued from the internal event queue and the HandleEvent() method will be called once per event.

```cpp
int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release ( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
code
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
code
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
code
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win32::Win32InputServer
Win32::Win32InputServer Class Reference

#include <win32inputserver.h>

Inheritance diagram for Win32::Win32InputServer:
Detailed Description

Win32-specific InputServer (provides a default Keyboard and Mouse).

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Win32InputServer()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~Win32InputServer()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>void Open()</code></td>
<td>Open the input server</td>
</tr>
<tr>
<td><code>void Close()</code></td>
<td>Close the input server</td>
</tr>
<tr>
<td><code>void OnFrame()</code></td>
<td>Call after processing window events (reads DInput raw mouse events)</td>
</tr>
<tr>
<td><code>void SetMaxNumLocalPlayers(SizeT maxNumLocalPlayers)</code></td>
<td>Set the max number of local players for this application (default is 4)</td>
</tr>
<tr>
<td><code>SizeT GetMaxNumLocalPlayers()</code> const</td>
<td>Get the max number of local players</td>
</tr>
<tr>
<td><code>bool IsOpen()</code></td>
<td>Return true if open</td>
</tr>
<tr>
<td><code>void SetQuitRequested(bool b)</code></td>
<td>Set the quit requested flag</td>
</tr>
<tr>
<td><code>bool IsQuitRequested()</code> const</td>
<td>Return true if some subsystem has requested to quit the app (e.g. Alt-F4)</td>
</tr>
<tr>
<td><code>void Reset()</code></td>
<td>Reset input state</td>
</tr>
<tr>
<td><code>const Ptr&lt;Input::Keyboard&gt; &amp; GetDefaultKeyboard()</code> const</td>
<td>Get the default keyboard input handler</td>
</tr>
<tr>
<td><code>const Ptr&lt;Input::Mouse&gt; &amp; GetDefaultMouse()</code> const</td>
<td>Get the default mouse input handler</td>
</tr>
<tr>
<td><code>Ptr&lt;Input::GamePad&gt; GetDefaultGamePad(IndexT playerIndex)</code> const</td>
<td>Get default gamepad handler (playerIndex is valid up to MaxNumLocalPlayers)</td>
</tr>
</tbody>
</table>
void AttachInputHandler (Input::InputPriority::Code pri, const Ptr< Input::InputHandler > &inputHandler)
attach an input handler

void RemoveInputHandler (const Ptr< Input::InputHandler > &inputHandler)
remove an input handler

virtual void BeginFrame ()
call before processing window events

void EndFrame ()
call at end of frame

void PutEvent (const Input::InputEvent &ie)
put an input event into the handler chain

void ClearMouseCapture ()
clear the current mouse capture (if exists)

void ClearKeyboardCapture ()
clear the current keyboard capture (if exists)

void ClearCapture ()
clear both mouse and keyboard captures

const Ptr< Input::InputHandler > & GetMouseCaptureHandler () const
return the current mouse capture input handler
(return invalid ptr if no capture set)

const Ptr< Input::InputHandler > & GetKeyboardCaptureHandler () const
return the current keyboard capture input handler
(return invalid ptr if no capture set)

void ObtainMouseCapture (const Ptr< Input::InputHandler > &inputHandler)
only call from InputHandler: capture mouse input to the given input handler

void ReleaseMouseCapture (const Ptr< Input::InputHandler > &inputHandler)
only call from InputHandler: release mouse capture

void ObtainKeyboardCapture (const Ptr< Input::InputHandler > &inputHandler)
only call from InputHandler: capture keyboard input to the given input handler
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void ReleaseKeyboardCapture (const Ptr&lt;Input::InputHandler&gt; &amp;inputHandler)</code></td>
<td>Only call from <code>InputHandler</code>: release keyboard capture</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>Decrease refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>Return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>Return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>Return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool OpenDInputMouse()</code></td>
<td>setup the DirectInput mouse device for tracking mouse movement</td>
</tr>
<tr>
<td><code>void CloseDInputMouse()</code></td>
<td>shutdown the DirectInput mouse device</td>
</tr>
<tr>
<td><code>void ReadDInputMouse()</code></td>
<td>get mouse readings</td>
</tr>
<tr>
<td><code>const Math::float2 &amp; GetMouseMovement()</code></td>
<td>get the current mouse movement</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool Win32::Win32InputServer::OpenDInputMouse() [protected]

setup the DirectInput mouse device for tracking mouse movement

This intitialies a DirectInput mouse device in order to track raw mouse movement (WM mouse events stop at the screen borders).

void Win32::Win32InputServer::CloseDInputMouse() [protected]

shutdown the DirectInput mouse device

Close the DirectInput mouse and DirectInput.

void Win32::Win32InputServer::ReadDInputMouse() [protected]

get mouse readings

Read data from the DirectInput mouse (relative mouse movement since the last frame).

void Base::InputServerBase::SetMaxNumLocalPlayers(SizeT num) [inherited]

set the max number of local players for this application (default is 4)

Setup the maximum number of local players for this application. The default number is 1. This defines the number of game pad objects created and queried.

void Base::InputServerBase::EndFrame() [inherited]

call at end of frame

Call this somewhere towards the end of frame, when it is guaranteed
that noone needs input anymore.

```cpp
void Base::InputServerBase::PutEvent (const Input::InputEvent & ie ) [inherited]
```

put an input event into the handler chain

NOTE: MouseMove and RawMouseMove events will be distributed to all input handlers regardless of mouse capture state!

```cpp
void Base::InputServerBase::ClearMouseCapture ( ) [inherited]
```

clear the current mouse capture (if exists)

This clears the currently set mouse capture (if exists).

```cpp
void Base::InputServerBase::ClearKeyboardCapture ( ) [inherited]
```

clear the current keyboard capture (if exists)

This clears the currently set keyboard capture (if exists).

```cpp
void Base::InputServerBase::ClearCapture ( ) [inherited]
```

clear both mouse and keyboard captures

This clears the mouse and keyboards captures, if set.

```cpp
void Base::InputServerBase::ObtainMouseCapture ( const Ptr< Input::InputHandler > & inputHandler ) [inherited]
```

only call from InputHandler: capture mouse input to the given input handler

Obtain the mouse capture. All mouse input will go exclusively to the capture input handler until ReleaseMouseCapture() is called.

```cpp
const Ptr<
```
void Base::InputServerBase::ReleaseMouseCapture(Input::InputHandler &inputHandler) [inherited]

only call from InputHandler: release mouse capture

Release the mouse capture. The provided pointer must match the current capture input handler.

void Base::InputServerBase::ObtainKeyboardCapture(const Ptr<Input::InputHandler> &inputHandler) [inherited]

only call from InputHandler: capture keyboard input to the given input handler

Obtain the keyboard capture. All keyboard input will go exclusively to the capture input handler until ReleaseKeyboardCapture() is called.

void Base::InputServerBase::ReleaseKeyboardCapture(const Ptr<Input::InputHandler> &inputHandler) [inherited]

only call from InputHandler: release keyboard capture

Release the mouse capture. The provided pointer must match the current capture input handler.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win32::Win32MiniDump
Win32::Win32MiniDump Class Reference

#include <win32minidump.h>

Inheritance diagram for Win32::Win32MiniDump:

```
Win32::Win32MiniDump

Debug::MiniDump
```
Detailed Description

Win32 implementation of MiniDump.

(C) 2007 Radon Labs GmbH
Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void Setup ()</td>
<td>setup the the Win32 exception callback hook</td>
</tr>
<tr>
<td>static bool WriteMiniDump ()</td>
<td>write a mini dump</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Win32::Win32MiniDump::Setup() [static]
```

setup the the **Win32** exception callback hook

This static method registers our own exception handler with Windows.

```cpp
bool Win32::Win32MiniDump::WriteMiniDump() [static]
```

write a mini dump

This method is called by `n_assert()` and `n_error()` to write out a minidump file.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win32::Win32Mouse
Win32::Win32Mouse Class Reference

#include <win32mouse.h>

Inheritance diagram for Win32::Win32Mouse:
Detailed Description

Overrides the default Mouse input device class and provides raw mouse movement data via DirectInput. This is necessary because Windows WM_MOUSEMOVE messages stop at the screen border.

(C) 2007 Radon Labs GmbH
Public Member Functions

const `Math::float2` & **GetMovement** () const
- get mouse movement

virtual void **BeginCapture** ()
- capture input to this event handler

virtual void **EndCapture** ()
- end input capturing to this event handler

bool **ButtonPressed** (`Input::MouseButton::Code` btn) const
- return true if button is currently pressed

bool **ButtonDown** (`Input::MouseButton::Code` btn) const
- return true if button was down at least once in current frame

bool **ButtonUp** (`Input::MouseButton::Code` btn) const
- return true if button was up at least once in current frame

bool **ButtonDoubleClicked** (`Input::MouseButton::Code` btn) const
- return true if a button has been double clicked

bool **WheelForward** () const
- return true if mouse wheel rotated forward

bool **WheelBackward** () const
- return true if mouse wheel rotated backward

const `Math::float2` & **GetPixelPosition** () const
- get current absolute mouse position (in pixels)

const `Math::float2` & **GetScreenPosition** () const
- get current screen space mouse position (0.0 .. 1.0)

bool **IsAttached** () const
- return true if the input handler is currently attached

bool **IsCapturing** () const
- return true if this input handler captures input

int **GetRefCount** () const
- get the current refcount

void **AddRef** ()
- increment refcount by one
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void Release ()</strong></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::String &amp;className) const</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><strong>bool IsA (const Rtti &amp;rtti) const</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><strong>bool IsA (const Util::String &amp;rttiName) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><strong>bool IsA (const Util::FourCC &amp;rttiFourCC) const</strong></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetClassName () const</strong></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC GetClassFourCC () const</strong></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void OnAttach ()</td>
<td>called when the handler is attached to the input server</td>
</tr>
<tr>
<td>virtual void OnBeginFrame ()</td>
<td>called on InputServer::BeginFrame()</td>
</tr>
<tr>
<td>virtual bool OnEvent (const Input::InputEvent &amp;inputEvent)</td>
<td>called when an input event should be processed</td>
</tr>
<tr>
<td>virtual void OnObtainCapture ()</td>
<td>called when input handler obtains capture</td>
</tr>
<tr>
<td>virtual void OnReleaseCapture ()</td>
<td>called when input handler looses capture</td>
</tr>
<tr>
<td>virtual void OnReset ()</td>
<td>reset the input handler</td>
</tr>
<tr>
<td>virtual void OnRemove ()</td>
<td>called when the handler is removed from the input server</td>
</tr>
<tr>
<td>virtual void OnEndFrame ()</td>
<td>called on InputServer::EndFrame;</td>
</tr>
</tbody>
</table>
Member Function Documentation

int Core::RefCounted::GetRefCount ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release ( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Win32::Win2SkinnedMeshRenderer
Win32::Win32SkinnedMeshRenderer
Class Reference

#include <win32skinnedmeshrenderer.h>

Inheritance diagram for Win32::Win32SkinnedMeshRenderer:
Detailed Description

D3D9/Xbox360 implemention of SkinnedMeshRenderer.

(C) 2008 Radon Labs GmbH
Public Types

typedef IndexT **DrawHandle

an abstract draw handle
### Public Member Functions

*Win32SkinnedMeshRenderer* ()

*constructor*

virtual *~Win32SkinnedMeshRenderer* ()

*destructor*

Characters::SkinningTechnique::Code *GetSkinningTechnique* () const

*get the skinning technique used by the renderer*

void *Setup* ()

*setup the renderer*

void *Discard* ()

*discard the renderer*

bool *IsValid* () const

*return true if renderer is valid*

int *GetRefCount* () const

*get the current refcount*

void *AddRef* ()

*increment refcount by one*

void *Release* ()

*decrement refcount and destroy object if refcount is zero*

bool *IsInstanceOf* (const *Rtti* &rtti) const

*return true if this object is instance of given class*

bool *IsInstanceOf* (const *Util::String* &className) const

*return true if this object is instance of given class by string*

bool *IsInstanceOf* (const *Util::FourCC* &classFourCC) const

*return true if this object is instance of given class by fourcc*

bool *IsA* (const *Rtti* &rtti) const

*return true if this object is instance of given class, or a derived class*

bool *IsA* (const *Util::String* &rttiName) const

*return true if this object is instance of given class,*
or a derived class, by string

<table>
<thead>
<tr>
<th>bool IsA (const Util::FourCC &amp;rttiFourCC) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp; GetClassName () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Util::FourCC GetClassFourCC () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>static void</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>OnBeginFrame()</code></td>
<td>call once at beginning of frame</td>
</tr>
<tr>
<td>void</td>
<td><code>OnEndFrame()</code></td>
<td>call once at end of frame (after rendering)</td>
</tr>
<tr>
<td>void</td>
<td><code>BeginGatherSkins()</code></td>
<td>begin gathering software-skinned meshes</td>
</tr>
<tr>
<td>DrawHandle</td>
<td><code>RegisterSoftwareSkinnedMesh</code></td>
<td>update a software skinned mesh</td>
</tr>
<tr>
<td>void</td>
<td><code>EndGatherSkins()</code></td>
<td>end gathering software-skinned meshes</td>
</tr>
<tr>
<td>void</td>
<td><code>UpdateSoftwareSkinnedMeshes()</code></td>
<td>update software-skinned meshes</td>
</tr>
<tr>
<td>void</td>
<td><code>DrawSoftwareSkinnedMesh</code></td>
<td>draw a software skinned mesh</td>
</tr>
<tr>
<td>void</td>
<td><code>DrawGPUTextureSkinnedMesh</code></td>
<td>draw a skinned mesh</td>
</tr>
<tr>
<td>IndexT</td>
<td><code>AllocJointTextureRow()</code></td>
<td>allocate a row index in the joint texture</td>
</tr>
<tr>
<td>void</td>
<td><code>FreeJointTextureRow</code></td>
<td>free a row index in the joint texture</td>
</tr>
<tr>
<td>void *</td>
<td><code>AcquireJointTextureRowPointer</code></td>
<td>get a pointer to the joint texture row for the given character instance</td>
</tr>
</tbody>
</table>
Member Function Documentation

**SkinnedMeshRendererBase::DrawHandle**

Base::SkinnedMeshRendererBase::RegisterSoftwareSkinnedMesh

\[
\text{const } \text{Ptr}<\text{Characters::CharacterInstance}> &\text{ } \text{& const } \text{Ptr}<\text{CoreGraphics::Mesh}> &\text{ }
\]

update a software skinned mesh

This method should only be called when RequiresSoftwareSkinning() returns true!

This registers a mesh for software-skinning in the **UpdateSoftwareSkinnedMeshes()** which must be called after **EndGatherSkins()**.

This method may be called more then once per character-instance/mesh combination! The method will drop duplicates.

**void**

Base::SkinnedMeshRendererBase::UpdateSoftwareSkinnedMeshes()

[protected, inherited]

update software-skinned meshes

On platforms with software-skinning, this method should perform the skinning for all meshes gathered during the GatherSkins pass.

**void**

Base::SkinnedMeshRendererBase::DrawSoftwareSkinnedMesh

\[
\text{(DrawHandle } h, \text{ IndexT } \text{primGroupIndex, })\]

draw a software skinned mesh

This method should only be called when RequiresSoftwareSkinning() returns true!
Software-skinning platforms call this method with the DrawHandle returned by UpdateSoftwareSkinnedMesh() to draw a portion of the skinned mesh:

```cpp
int Core::RefCounted::GetRefCount () const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef () [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release () [inline, inherited]
```

decremented refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName () const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC () const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks () [static, inherited]
```
dump ref counting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Win32::Win32StringConverter
Win32::Win32StringConverter Class Reference

#include <win32stringconverter.h>
Detailed Description

Convert between UTF-8 and 16-bit wide strings.

(C) 2009 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTF8ToWide (const Util::String &amp;src, ushort *dst, SizeT dstMaxBytes)</td>
<td>convert from UTF-8 encoded string object to wide string, return number of used bytes</td>
</tr>
<tr>
<td>UTF8ToWide (const char *src, ushort *dst, SizeT dstMaxBytes)</td>
<td>convert from UTF-8 raw string to wide string, return number of used bytes</td>
</tr>
<tr>
<td>WideToUTF8 (ushort *src)</td>
<td>convert from wide string to UTF-8 string</td>
</tr>
</tbody>
</table>
Win32::Win32SystemInfo
Win32::Win32SystemInfo Class Reference

#include <win32systeminfo.h>

Inheritance diagram for Win32::Win32SystemInfo:
Detailed Description

Provide information about the system we're running on.

(C) 2008 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>host platforms</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th>CpuType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>CPU types</em></td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Class</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win32SystemInfo</td>
<td><code>Win32SystemInfo()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>Platform</td>
<td><code>GetPlatform()</code> const</td>
<td>get host platform</td>
</tr>
<tr>
<td>CpuType</td>
<td><code>GetCpuType()</code> const</td>
<td>get cpu type</td>
</tr>
<tr>
<td>SizeT</td>
<td><code>GetNumCpuCores()</code> const</td>
<td>get number of processors</td>
</tr>
<tr>
<td>SizeT</td>
<td><code>GetPageSize()</code> const</td>
<td>get page size</td>
</tr>
</tbody>
</table>
Static Public Member Functions

```cpp
static Util::String PlatformAsString (Platform p)  
convert platform to string

static Util::String CpuTypeAsString (CpuType cpu)  
convert CpuType to string
```

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:50 2010
Win360::D3D9IndexBuffer
Win360::D3D9IndexBuffer Class Reference

#include <d3d9indexbuffer.h>

Inheritance diagram for Win360::D3D9IndexBuffer:
Detailed Description

D3D9/Xbox360 implementation of index buffer.

FIXME: need to handle DeviceLost render event!

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>Usage</strong></th>
<th>resource usage flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><strong>State</strong></td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9IndexBuffer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~D3D9IndexBuffer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>virtual void Unload ()</strong></td>
<td>Unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td><strong>void * Map (MapType mapType)</strong></td>
<td>Map index buffer for CPU access</td>
</tr>
<tr>
<td><strong>void Unmap ()</strong></td>
<td>Unmap the resource</td>
</tr>
<tr>
<td>*<em>void SetD3D9IndexBuffer (IDirect3DIndexBuffer9 <em>ptr)</em></em></td>
<td>Set D3D9 index buffer pointer</td>
</tr>
<tr>
<td><strong>IDirect3DIndexBuffer9 * GetD3D9IndexBuffer () const</strong></td>
<td>Get D3D9 index buffer pointer</td>
</tr>
<tr>
<td><strong>void SetIndexType (CoreGraphics::IndexType::Code type)</strong></td>
<td>Set the index type (Index16 or Index32)</td>
</tr>
<tr>
<td><strong>CoreGraphics::IndexType::Code GetIndexType () const</strong></td>
<td>Get the index type (Index16 or Index32)</td>
</tr>
<tr>
<td><strong>void SetNumIndices (SizeT num)</strong></td>
<td>Set number of indices</td>
</tr>
<tr>
<td><strong>SizeT GetNumIndices () const</strong></td>
<td>Get number of indices</td>
</tr>
<tr>
<td><strong>void SetUsage (Usage usage)</strong></td>
<td>Set resource usage type</td>
</tr>
<tr>
<td><strong>Usage GetUsage () const</strong></td>
<td>Get resource usage type</td>
</tr>
<tr>
<td><strong>void SetAccess (Access access)</strong></td>
<td>Set resource cpu access type</td>
</tr>
<tr>
<td><strong>Access GetAccess () const</strong></td>
<td>Get cpu access type</td>
</tr>
<tr>
<td><strong>void SetAsyncEnabled (bool b)</strong></td>
<td>Set async enabled</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsAsyncEnabled () const</code></td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>Lock ()</code></td>
<td>set locked to true</td>
</tr>
<tr>
<td><code>Unlock ()</code></td>
<td>set locked to false</td>
</tr>
<tr>
<td><code>IsLocked () const</code></td>
<td>returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td><code>SetResourceId (const ResourceId &amp;id)</code></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><code>GetResourceId () const</code></td>
<td>get the resource identifier</td>
</tr>
<tr>
<td><code>SetLoader (const Ptr&lt;ResourceLoader&gt; &amp;loader)</code></td>
<td>set optional resource loader</td>
</tr>
<tr>
<td><code>GetLoader () const</code></td>
<td>get optional resource loader</td>
</tr>
<tr>
<td><code>SetSaver (const Ptr&lt;ResourceSaver&gt; &amp;saver)</code></td>
<td>set optional resource saver</td>
</tr>
<tr>
<td><code>GetSaver () const</code></td>
<td>get optional resource saver</td>
</tr>
<tr>
<td><code>GetUseCount () const</code></td>
<td>get current use count</td>
</tr>
<tr>
<td><code>Load ()</code></td>
<td>load the resource</td>
</tr>
<tr>
<td><code>SetState (State s)</code></td>
<td>set current state (usually only called during Load())</td>
</tr>
<tr>
<td><code>GetState () const</code></td>
<td>get current state</td>
</tr>
<tr>
<td><code>IsLoaded () const</code></td>
<td>return true if current state is Loaded</td>
</tr>
<tr>
<td><code>IsPending () const</code></td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td>Data Type</td>
<td>Function</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
</tr>
<tr>
<td>int</td>
<td>GetRefCount() const</td>
</tr>
<tr>
<td>void</td>
<td>AddRef()</td>
</tr>
<tr>
<td>void</td>
<td>Release()</td>
</tr>
<tr>
<td>bool</td>
<td>LoadFailed() const</td>
</tr>
<tr>
<td>virtual bool</td>
<td>Save()</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf(const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf(const Util::String &amp;className) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf(const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsA(const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsA(const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsA(const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName() const</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC() const</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
<tr>
<td>Protected Member Functions</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td>void <strong>IncrUseCount</strong> ()</td>
<td></td>
</tr>
<tr>
<td><em>increment use count</em></td>
<td></td>
</tr>
<tr>
<td>void <strong>DecrUseCount</strong> ()</td>
<td></td>
</tr>
<tr>
<td><em>decrement use count</em></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

**Resource::State**

**Resources::Resource::Load** ( ) [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the **IsPending()** method will return true as long as the load is in progress, and **IsLoaded()** will become true when the loading process has finished. If the load has failed, **IsPending()** will switch to false and **IsLoaded()** will not be true.

**bool**

**Resources::Resource::Save** ( ) [virtual, inherited]

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

**int**

**Core::RefCounted::GetRefCount** ( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

**void**

**Core::RefCounted::AddRef** ( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

**void**

**Core::RefCounted::Release** ( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC
Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void
Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Win360::D3D9MemoryIndexBufferLoader
Win360::D3D9MemoryIndexBufferLoader

Class Reference

#include <d3d9memoryindexbufferloader.h>

Inheritance diagram for Win360::D3D9MemoryIndexBufferLoader:
Detailed Description

Initialize a `D3D9IndexBuffer` from data in memory for the Win32/Xbox360 platform. This resource loader only creates static IndexBuffers which are initialized once and are not accessible by the CPU.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool <strong>OnLoadRequested ()</strong></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td>void *<em>Setup (CoreGraphics::IndexType::Code indexType, SizeT numIndices, void <em>indexDataPtr, SizeT indexDataSize, CoreGraphics::IndexBuffer::Usage usage=CoreGraphics::IndexBuffer::UsageImmutable, CoreGraphics::IndexBuffer::Access access=CoreGraphics::IndexBuffer::AccessNone)</em></em></td>
<td>setup index buffer from existing data, or provide 0 pointer if empty index buffer should be created</td>
</tr>
<tr>
<td>virtual void <strong>OnAttachToResource (const Ptr&lt;Resource&gt; &amp;res)</strong></td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>virtual void <strong>OnRemoveFromResource ()</strong></td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>bool <strong>IsAttachedToResource ()</strong> const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt;Resource&gt; &amp; <strong>GetResource ()</strong> const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td>virtual bool <strong>CanLoadAsync ()</strong> const</td>
<td>return true if asynchronous loading is supported</td>
</tr>
<tr>
<td>virtual void <strong>OnLoadCancelled ()</strong></td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td>virtual bool <strong>OnPending ()</strong></td>
<td>call frequently while after <strong>OnLoadRequested ()</strong> to put Resource into loaded state</td>
</tr>
<tr>
<td><strong>Resource::State GetState ()</strong> const</td>
<td>return current state</td>
</tr>
<tr>
<td>virtual void <strong>Reset ()</strong></td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td>int <strong>GetRefCount ()</strong> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>increment refcount by one</td>
<td></td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

void SetState (Resource::State S)

set current state
Member Function Documentation

bool Win360::D3D9MemoryIndexBufferLoader::OnLoadRequested () [virtual]
called by resource when a load is requested

This will create a D3D9 IndexBuffer using the data provided by our Setup() method and set our resource object (which must be a D3D9IndexBuffer object).

Reimplemented from Resources::ResourceLoader.

bool Resources::ResourceLoader::CanLoadAsync () const [virtual, inherited]
return true if asynchronous loading is supported

This method should be overridden in a subclass and indicates whether the resource loader supports asynchronous resource loading. If asynchronous loading is requested, the OnLoadRequested() method will return immediately and the Resource object will be put into Pending state. Afterwards, the Resource object needs to poll the ResourceLoader using the OnPending method, which will eventually setup the Resource object.

Reimplemented in Direct3D9::D3D9StreamShaderLoader, Models::StreamModelLoader, and Resources::StreamResourceLoader.

void Resources::ResourceLoader::OnLoadCancelled () [virtual, inherited]
called by resource to cancel a pending load

This method is called by our Resource object if a pending asynchronous load should be cancelled.

Reimplemented in Models::StreamModelLoader, and
Resources::StreamResourceLoader.

bool Resources::ResourceLoader::OnPending() [virtual, inherited]

call frequently while after OnLoadRequested() to put Resource into
loaded state

This method should be called at some time after OnLoadRequested() as long as the ResourceLoader is in the Pending state. This will check whether the asynchronous loader job has finished, and if yes, setup the Resource object, bringing it from the Pending into the Loaded state. If something goes wrong, the ResourceLoader will go into the Failed state. If the outstanding loader job isn't finished yet, the ResourceLoader should remain in Pending state, and the method should return false. Otherwise the Resource should be initialized, and the method should return true.

Reimplemented in Models::StreamModelLoader, and Resources::StreamResourceLoader.

int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String & Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Win360::D3D9MemoryVertexBufferLoader
Win360::D3D9MemoryVertexBufferLoader
Class Reference

#include <d3d9memoryvertexbufferloader.h>

Inheritance diagram for Win360::D3D9MemoryVertexBufferLoader:
Detailed Description

Initialize a D3D9VertexBuffer from data in memory on the Win32/Xbox360 platform. This resource loader only creates static VertexBuffers which are initialized once and are not accessible by the CPU.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool bool OnLoadRequested ()</td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td>virtual void void Setup (const Util::Array&lt; CoreGraphics::VertexComponent &gt; &amp;vertexComponents, SizeT numVertices, void *vertexDataPtr, SizeT vertexDataSize, CoreGraphics::VertexBuffer::Usage usage, CoreGraphics::VertexBuffer::Access access)</td>
<td>setup vertex buffer data, must remain valid until OnLoadRequested() is called!</td>
</tr>
<tr>
<td>virtual void void OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>virtual void void OnRemoveFromResource ()</td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>bool IsAttachedToResource () const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetResource () const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td>virtual bool CanLoadAsync () const</td>
<td>return true if asynchronous loading is supported</td>
</tr>
<tr>
<td>virtual void void OnLoadCancelled ()</td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td>virtual bool OnPending ()</td>
<td>call frequently while after OnLoadRequested() to put Resource into loaded state</td>
</tr>
<tr>
<td>Resource::State GetState () const</td>
<td>return current state</td>
</tr>
<tr>
<td>virtual void void Reset ()</td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>void</td>
<td><strong>Release</strong> ()</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
void SetState (Resource::State S)  
  set current state
```
Member Function Documentation

```cpp
bool Win360::D3D9MemoryVertexBufferLoader::OnLoadRequested() [virtual]
```
called by resource when a load is requested

This will create a D3D9 vertex buffer and vertex declaration object from the data provided in the `Setup()` method and setup our resource object (which must be a `D3D9VertexBuffer` object).

Reimplemented from `Resources::ResourceLoader`.

```cpp
bool Resources::ResourceLoader::CanLoadAsync() const [virtual, inherited]
```
return true if asynchronous loading is supported

This method should be overriden in a subclass and indicates whether the resource loader supports asynchronous resource loading. If asynchronous loading is requested, the `OnLoadRequested()` method will return immediately and the `Resource` object will be put into Pending state. Afterwards, the `Resource` object needs to poll the `ResourceLoader` using the `OnPending` method, which will eventually setup the `Resource` object.

Reimplemented in `Direct3D9::D3D9StreamShaderLoader`, `Models::StreamModelLoader`, and `Resources::StreamResourceLoader`.

```cpp
void Resources::ResourceLoader::OnLoadCancelled() [virtual, inherited]
```
called by resource to cancel a pending load

This method is called by our `Resource` object if a pending asynchronous load should be cancelled.

Reimplemented in `Models::StreamModelLoader`, and
Resources::StreamResourceLoader.

bool
Resources::ResourceLoader::OnPending() [virtual, inherited]
call frequently while after OnLoadRequested() to put Resource into loaded state

This method should be called at some time after OnLoadRequested() as long as the ResourceLoader is in the Pending state. This will check whether the asynchronous loader job has finished, and if yes, setup the Resource object, bringing it from the Pending into the Loaded state. If something goes wrong, the ResourceLoader will go into the Failed state. If the outstanding loader job isn't finished yet, the ResourceLoader should remain in Pending state, and the method should return false. Otherwise the Resource should be initialized, and the method should return true.

Reimplemented in Models::StreamModelLoader, and Resources::StreamResourceLoader.

int
Core::RefCounted::GetRefCount() const [inline, inherited]
get the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef() [inline, inherited]
increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release() [inline, inherited]
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
const Util::String & Core::RefCounted::GetClassName () const [inline, inherited]

get the class name

Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC () const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks () [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:51 2010
Win360::D3D9ShapeRenderer
Win360::D3D9ShapeRenderer Class Reference

#include <d3d9shaperenderer.h>

Inheritance diagram for Win360::D3D9ShapeRenderer:
Detailed Description

D3D9/Xbox360 implementation of ShapeRenderer.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| **D3D9ShapeRenderer** () | *constructor*
| `virtual ~D3D9ShapeRenderer` () | *destructor*
<p>| <code>void Open</code> () | <code>open the shape renderer</code> |
| <code>void Close</code> () | <code>close the shape renderer</code> |
| <code>void DrawShapes</code> () | <code>draw attached shapes and clear deferred stack, must be called inside render loop</code> |
| <code>bool IsOpen</code> () const | <code>return true if open</code> |
| <code>void DeleteShapesByThreadId</code> (Threading::ThreadId threadId) | <code>delete shapes of given thread id</code> |
| <code>void AddShape</code> (const CoreGraphics::RenderShape &amp;shape) | <code>add a shape for deferred rendering (can be called from outside render loop)</code> |
| <code>void AddShapes</code> (const Util::Array<a href="">CoreGraphics::RenderShape</a> &amp;shapeArray) | <code>add multiple shapes</code> |
| <code>void AddWireFrameBox</code> (const Math::bbox &amp;boundingBox, const Math::float4 &amp;color, Threading::ThreadId threadId) | <code>add wireframe box</code> |
| <code>int GetRefCount</code> () const | <code>get the current refcount</code> |
| <code>void AddRef</code> () | <code>increment refcount by one</code> |
| <code>void Release</code> () | <code>decrement refcount and destroy object if refcount is zero</code> |
| <code>bool IsInstanceOf</code> (const Rtti &amp;rtti) const | |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName() const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC() const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]

get the current refcount

Return the current refcount of the object.
```

```cpp
void Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.
```

```cpp
void Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.
```

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]

get the class name

Get the class name of the object.
```

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.
```

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Win360::D3D9StreamMeshLoader
Win360::D3D9StreamMeshLoader
Class Reference

#include <d3d9streammeshloader.h>

Inheritance diagram for Win360::D3D9StreamMeshLoader:
Detailed Description

Setup a Mesh object from a stream. Supports the following file formats:

- nvx2 (Nebula2 binary mesh file format)
- nvx3 (Nebula3 binary mesh file format)
- n3d3 (Nebula3 ascii mesh file format)

Todo:
- document file formats

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9StreamMeshLoader</strong> (constructor)</td>
<td></td>
</tr>
<tr>
<td><strong>void SetUsage(Base::ResourceBase::Usage usage)</strong></td>
<td>set the intended resource usage (default is UsageImmutable)</td>
</tr>
<tr>
<td><strong>Base::ResourceBase::Usage GetUsage() const</strong></td>
<td>get resource usage</td>
</tr>
<tr>
<td><strong>void SetAccess(Base::ResourceBase::Access access)</strong></td>
<td>set the intended resource access (default is AccessNone)</td>
</tr>
<tr>
<td><strong>Base::ResourceBase::Access GetAccess() const</strong></td>
<td>get the resource access</td>
</tr>
<tr>
<td><strong>virtual bool CanLoadAsync() const</strong></td>
<td>override in subclass: return true if asynchronous loading is supported (default is true)</td>
</tr>
<tr>
<td><strong>virtual bool OnLoadRequested()</strong></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td><strong>virtual void OnLoadCancelled()</strong></td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td><strong>virtual bool OnPending()</strong></td>
<td>call frequently while after OnLoadRequested() to put Resource into loaded state</td>
</tr>
<tr>
<td><strong>virtual void OnAttachToResource(const Ptr&lt;Resource&gt; &amp;res)</strong></td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td><strong>virtual void OnRemoveFromResource()</strong></td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td><strong>bool IsAttachedToResource() const</strong></td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td><strong>const Ptr&lt;Resource&gt; &amp;</strong> GetResource() const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td><strong>Resource::State GetState() const</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>virtual void Reset()</td>
<td>resets loader-stats e.g. state</td>
</tr>
<tr>
<td>int GetRefCount() const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool IsA (const Util::String &amp;rttiName) const</td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td>bool IsA (const Util::FourCC &amp;rttiFourCC) const</td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName() const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC() const</td>
<td>get the class FourCC code</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>DumpRefCountingLeaks()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetState</strong> <em>(Resource::State S)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set current state</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool Resources::StreamResourceLoader::CanLoadAsync() const [virtual, inherited]
```

override in subclass: return true if asynchronous loading is supported (default is true)

Indicate whether this resource loader supports asynchronous loading. The default is true. Override this method in a subclass and return false otherwise.

Reimplemented from `Resources::ResourceLoader`.

Reimplemented in `Direct3D9::D3D9StreamShaderLoader`.

```cpp
bool Resources::StreamResourceLoader::OnLoadRequested() [virtual, inherited]
```

called by resource when a load is requested

Handle the generic load request. In the asynchronous case, this method will fire a ReadStream message and go into pending state. The client will then call `OnPending()` periodically which checks whether the ReadStream message has been handled and continue accordingly. In the synchronous state the method will create an `IO::Stream` object and call the `SetupResourceFromStream()` method directly.

Reimplemented from `Resources::ResourceLoader`.

Reimplemented in `Resources::D3D9TextureStreamer`.

```cpp
void Resources::StreamResourceLoader::OnLoadCancelled() [virtual, inherited]
```

called by resource to cancel a pending load

This method is called when the currently pending asynchronous load
request should be cancelled.

Reimplemented from Resources::ResourceLoader.

bool
Resources::StreamResourceLoader::OnPending() [virtual, inherited]

call frequently while after OnLoadRequested() to put Resource into loaded state

This method is called periodically by the client when the resource is in pending state. The pending ReadStream message will be checked, and if it has been handled successfully the SetupResourceFromStream() method will be called and the Resource will go into Loaded state. If anything goes wrong, the resource will go into Failed state.

Reimplemented from Resources::ResourceLoader.

int
Core::RefCounted::GetRefCount() const [inline, inherited]

going the current refcount

Return the current refcount of the object.

void
Core::RefCounted::AddRef() [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void
Core::RefCounted::Release() [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String &
Core::RefCounted::GetClassName() const [inline, inherited]
get the class name
Get the class name of the object.

`Util::FourCC` Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code
Get the class FourCC of the object.

`void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]`

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Win360::D3D9StreamTextureSaver
Win360::D3D9StreamTextureSaver
Class Reference

#include <d3d9streamtexturesaver.h>

Inheritance diagram for Win360::D3D9StreamTextureSaver:
Detailed Description

D3D9/Xbox360 implementation of StreamTextureSaver.

(C) 2007 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual bool OnSave();</code></td>
<td>Called by resource when a save is requested</td>
</tr>
<tr>
<td><code>void SetStream(const Ptr&lt;IO::Stream&gt;&amp; stream);</code></td>
<td>Set stream to save to</td>
</tr>
<tr>
<td><code>const Ptr&lt;IO::Stream&gt;&amp; GetStream(); const</code></td>
<td>Get save-stream</td>
</tr>
<tr>
<td><code>void SetFormat(CoreGraphics::ImageFileFormat::Code fmt);</code></td>
<td>Set file format (default is JPG)</td>
</tr>
<tr>
<td><code>CoreGraphics::ImageFileFormat::Code GetFormat(); const</code></td>
<td>Get file format</td>
</tr>
<tr>
<td><code>void SetMipLevel(IndexT mipLevel);</code></td>
<td>Set the mip level to save (default is 0, the top level)</td>
</tr>
<tr>
<td><code>IndexT GetMipLevel(); const</code></td>
<td>Get the mip level to save</td>
</tr>
<tr>
<td><code>virtual void OnAttachToResource(const Ptr&lt;Resource&gt;&amp; res);</code></td>
<td>Called when the resource saver is attached to its resource</td>
</tr>
<tr>
<td><code>virtual void OnRemoveFromResource();</code></td>
<td>Called when the resource saver is removed from its resource</td>
</tr>
<tr>
<td><code>bool IsAttachedToResource(); const</code></td>
<td>Return true if attached to resource</td>
</tr>
<tr>
<td><code>const Ptr&lt;Resource&gt;&amp; GetResource(); const</code></td>
<td>Get pointer to resource</td>
</tr>
<tr>
<td><code>int GetRefCount(); const</code></td>
<td>Get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef();</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>void Release();</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>Function</td>
<td>Signature</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String &amp;className) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td>bool</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
Member Function Documentation

int Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount

Return the current refcount of the object.

void Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one

Increment the refcount of the object.

void Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name

Get the class name of the object.

const Util::FourCC & Core::RefCounted::GetClassFourCC( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Win360::D3D9TransformDevice
Win360::D3D9TransformDevice Class Reference

#include <d3d9transformdevice.h>

Inheritance diagram for Win360::D3D9TransformDevice:
Detailed Description

D3D9/Xbox360 version of TransformDevice.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>D3D9TransformDevice ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~D3D9TransformDevice ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>bool Open ()</code></td>
<td>Opens the transform device</td>
</tr>
<tr>
<td><code>void Close ()</code></td>
<td>Closes the transform device</td>
</tr>
<tr>
<td><code>void ApplyViewSettings ()</code></td>
<td>Updates shared shader variables dependent on view matrix</td>
</tr>
<tr>
<td>void <code>ApplyModelTransforms (const Ptr&lt;CoreGraphics::ShaderInstance&gt; &amp;shdInst)</code></td>
<td>Applies any model transform needed, implementation is platform dependend</td>
</tr>
<tr>
<td><code>bool isOpen () const</code></td>
<td>Returns true if device is open</td>
</tr>
<tr>
<td><code>void SetProjTransform (const Math::matrix44 &amp;m)</code></td>
<td>Sets projection transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetProjTransform ()</code></td>
<td>Gets current projection matrix</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetInvProjTransform ()</code></td>
<td>Gets inverted projection transform</td>
</tr>
<tr>
<td><code>void SetViewTransform (const Math::matrix44 &amp;m)</code></td>
<td>Sets view transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetViewTransform ()</code></td>
<td>Gets view transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetInvViewTransform ()</code></td>
<td>Gets current inverted view transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetViewProjTransform ()</code></td>
<td>Gets current view-projection transform</td>
</tr>
<tr>
<td><code>void SetModelTransform (const Math::matrix44 &amp;m)</code></td>
<td>Sets model transform</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetModelTransform ()</code></td>
<td>Gets model transform</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetInvModelTransform()</code></td>
<td>get current inverted model transform</td>
</tr>
<tr>
<td><code>GetModelViewTransform()</code></td>
<td>get current model-view matrix</td>
</tr>
<tr>
<td><code>GetInvModelViewTransform()</code></td>
<td>get current inverted model-view-transform</td>
</tr>
<tr>
<td><code>GetModelViewProjTransform()</code></td>
<td>get current model-view-projection transform</td>
</tr>
<tr>
<td><code>SetFocalLength(const Math::float2 &amp;len)</code></td>
<td>set focal length</td>
</tr>
<tr>
<td><code>GetFocalLength()</code></td>
<td>get focal length</td>
</tr>
<tr>
<td><code>GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti)</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rttiName)</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName()</code></td>
<td>get the class name</td>
</tr>
</tbody>
</table>
Util::FourCC

GetClassFourCC () const

get the class FourCC code
<table>
<thead>
<tr>
<th>static void</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
This method should be called as the very last before an application exits.
Win360::D3D9Types
Win360::D3D9Types Class Reference

#include <d3d9types.h>
Detailed Description

Provides static helper functions to convert from and to Direct3D data types and enumerations.

(C) 2007 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| static `D3DFORMAT` `AsD3DF9PixelFormat`
  `CoreGraphics::PixelFormat::Code p` | convert Nebula pixel format to D3D9 pixel format                                             |
| static `CoreGraphics::PixelFormat::Code` `AsNebulaPixelFormat`
  `D3DFORMAT f` | convert Direct3D to Nebula pixel format                                                      |
| static `D3DDECLTYPE` `AsD3D9VertexDeclarationType`
  `CoreGraphics::VertexComponent::Format f` | convert vertex component type to D3D9 declaration type                                       |
| static `D3DDECLUSAGE` `AsD3D9VertexDeclarationUsage`
  `CoreGraphics::VertexComponent::SemanticName n` | convert vertex component semantic name as D3D9 declaration usage                            |
| static `D3DPRIMITIVETYPE` `AsD3D9PrimitiveType`
  `CoreGraphics::PrimitiveTopology::Code` | convert primitive topology to D3D                                                             |
| static `D3DMULTISAMPLE_TYPE` `AsD3D9MultiSampleType`
  `CoreGraphics::AntiAliasQuality::Code` | convert antialias quality to D3D multisample type                                             |
| static `D3DXIMAGE_FILEFORMAT` `AsD3DXImageFileFormat`
  `CoreGraphics::ImageFileFormat::Code` | convert image file format to D3DX file format                                                 |
| static `D3DPOOL` `AsD3D9Pool`
  `Base::ResourceBase::Usage`, `Base::ResourceBase::Access` | convert Nebula3 resource usage/access flag pair into D3D flag pair                          |
| static `DWORD` `AsD3D9Usage`
  `Base::ResourceBase::Usage`, `Base::ResourceBase::Access` | convert Nebula3 resource usage/access flag pair into D3D usage/access flag pair            |
| static `D3DFORMAT` `IndexTypeAsD3D9Format`
  `CoreGraphics::IndexType::Code` | convert index type to D3DFORMAT                                                              |
Win360::D3D9VertexBuffer
Win360::D3D9VertexBuffer Class Reference

#include <d3d9vertexbuffer.h>

Inheritance diagram for Win360::D3D9VertexBuffer:

- Core::RefCounted
- Resources::Resource
- Base::ResourceBase
- Base::VertexBufferBase
- Win360::D3D9VertexBuffer
- CoreGraphics::VertexBuffer
Detailed Description

D3D9/Xbox360 implementation of VertexBuffer.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th></th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>resource usage flags</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>State</td>
</tr>
<tr>
<td></td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9VertexBuffer (const std::shared_ptr<a href="">CoreGraphics::VertexLayout</a>&amp; vertexLayout)</strong></td>
<td>Set vertex layout (set by resource loader)</td>
</tr>
<tr>
<td><strong>D3D9VertexBuffer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~D3D9VertexBuffer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>void Unload ()</strong></td>
<td>Unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td><em><em>void</em> Map (MapType mapType)</em>*</td>
<td>Map the vertices for CPU access</td>
</tr>
<tr>
<td><strong>void Unmap ()</strong></td>
<td>Unmap the resource</td>
</tr>
<tr>
<td><em><em>void SetD3D9VertexBuffer (IDirect3DVertexBuffer9</em> ptr)</em>*</td>
<td>Set D3D9 vertex buffer pointer</td>
</tr>
<tr>
<td><em><em>IDirect3DVertexBuffer9</em> GetD3D9VertexBuffer () const</em>*</td>
<td>Get pointer to D3D9 vertex buffer object</td>
</tr>
<tr>
<td><strong>void SetVertexLayout (const std::shared_ptr<a href="">CoreGraphics::VertexLayout</a>&amp; vertexLayout)</strong></td>
<td>Set vertex layout (set by resource loader)</td>
</tr>
<tr>
<td><strong>const std::shared_ptr<a href="">CoreGraphics::VertexLayout</a>&amp; GetVertexLayout () const</strong></td>
<td>Get the vertex layout</td>
</tr>
<tr>
<td><strong>void SetNumVertices (SizeT numVertices)</strong></td>
<td>Set number of vertices (set by resource loader)</td>
</tr>
<tr>
<td><strong>SizeT GetNumVertices () const</strong></td>
<td>Get number of vertices in the buffer</td>
</tr>
<tr>
<td><strong>void SetUsage (Usage usage)</strong></td>
<td>Set resource usage type</td>
</tr>
<tr>
<td><strong>Usage GetUsage () const</strong></td>
<td>Get resource usage type</td>
</tr>
<tr>
<td><strong>void SetAccess (Access access)</strong></td>
<td>Set resource CPU access type</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>GetAccess</strong> () const</td>
<td>get cpu access type</td>
</tr>
<tr>
<td><strong>SetAsyncEnabled</strong> (bool b)</td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td><strong>IsAsyncEnabled</strong> () const</td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><strong>Lock</strong> ()</td>
<td>set locked to true</td>
</tr>
<tr>
<td><strong>Unlock</strong> ()</td>
<td>set locked to false</td>
</tr>
<tr>
<td><strong>IsLocked</strong> () const</td>
<td>returns true if resource will be used as source for copy process soon</td>
</tr>
<tr>
<td><strong>SetResourceId</strong> (const ResourceId &amp;id)</td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><strong>GetResourceId</strong> () const</td>
<td>get the resource identifier</td>
</tr>
<tr>
<td><strong>SetLoader</strong> (const Ptr&lt;ResourceLoader&gt; &amp;loader)</td>
<td>set optional resource loader</td>
</tr>
<tr>
<td><strong>GetLoader</strong> () const</td>
<td>get optional resource loader</td>
</tr>
<tr>
<td><strong>SetSaver</strong> (const Ptr&lt;ResourceSaver&gt; &amp;saver)</td>
<td>set optional resource saver</td>
</tr>
<tr>
<td><strong>GetSaver</strong> () const</td>
<td>get optional resource saver</td>
</tr>
<tr>
<td><strong>GetUseCount</strong> () const</td>
<td>get current use count</td>
</tr>
<tr>
<td><strong>Load</strong> ()</td>
<td>load the resource</td>
</tr>
<tr>
<td><strong>SetState</strong> (State s)</td>
<td>set current state (usually only called during Load()!)</td>
</tr>
<tr>
<td><strong>GetState</strong> () const</td>
<td></td>
</tr>
</tbody>
</table>
get current state

bool IsLoaded () const
return true if current state is Loaded

bool IsPending () const
return true if current state is Pending

bool LoadFailed () const
return true if current state is Failed

virtual bool Save ()
save the resource

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
return true if this object is instance of given class, or a derived class, by fourcc
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const <code>Util::String &amp;</code> GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC</code> GetClassFourCC () const</td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

```cpp
static void DumpRefCountingLeaks ()
   dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
```
### Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><strong>IncrUseCount</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>increment use count</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>DecrUseCount</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Resource::State**

Resources::Resource::Load() [virtual, inherited]

load the resource

This loads the resource through the attached resource loader. Depending on the resource loader, the resource may happen synchronously or asynchronously. If the resource is loaded asynchronously, the `IsPending()` method will return true as long as the load is in progress, and `IsLoaded()` will become true when the loading process has finished. If the load has failed, `IsPending()` will switch to false and `IsLoaded()` will not be true.

```cpp
bool Resources::Resource::Save() [virtual, inherited]
```

save the resource

This will save the resource. A resource saver must be attached to the resource and the resource must be loaded for the method to succeed. Saving will always be performed synchronously.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

const Util::String
Core::RefCounted::GetClassName()

get the class name

Get the class name of the object.

Util::FourCC
Core::RefCounted::GetClassFourCC()

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Win360::D3D9VertexLayout
Win360::D3D9VertexLayout Class Reference

#include <d3d9vertexlayout.h>

Inheritance diagram for Win360::D3D9VertexLayout:
Detailed Description

D3D9/Xbox360-implementation of vertex layout.

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>D3D9VertexLayout()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~D3D9VertexLayout()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>Setup (const Util::Array&lt;CoreGraphics::VertexComponent&gt; setup)</code></td>
<td>setup the vertex layout</td>
</tr>
<tr>
<td><code>Discard()</code></td>
<td>discard the vertex layout object</td>
</tr>
<tr>
<td><code>SetD3D9VertexDeclaration (IDirect3DVertexDeclaration9 * ptr)</code></td>
<td>set d3d9 vertex declaration pointer</td>
</tr>
<tr>
<td><code>GetD3D9VertexDeclaration ()</code></td>
<td>get pointer to d3d9 vertex declaration</td>
</tr>
<tr>
<td><code>IsValid () const</code></td>
<td>return true if valid has been setup</td>
</tr>
<tr>
<td><code>GetNumComponents () const</code></td>
<td>get number of components</td>
</tr>
<tr>
<td><code>GetComponentAt (IndexT i)</code></td>
<td>get vertex component at index</td>
</tr>
<tr>
<td><code>HasComponent (CoreGraphics::VertexComponent::SemanticName semName, IndexT semIndex)</code></td>
<td>return true if vertex component exists</td>
</tr>
<tr>
<td><code>FindComponent (CoreGraphics::VertexComponent::SemanticName semName, IndexT semIndex)</code></td>
<td>get index of vertex component by semantics</td>
</tr>
<tr>
<td><code>GetVertexByteSize () const</code></td>
<td>get the vertex stride in number of bytes</td>
</tr>
<tr>
<td><code>GetVertexComponents ()</code></td>
<td>get vertex components</td>
</tr>
<tr>
<td><code>GetRefCount () const</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>Decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(const Util::String &amp;rtti)</code></td>
<td>Return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti)</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsA(const Util::String &amp;rtti)</code></td>
<td>Return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>IsA(const Util::FourCC &amp;rtti)</code></td>
<td>Return true if this object is instance of given class, by fourcc</td>
</tr>
<tr>
<td><code>GetClassName()</code></td>
<td>Get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC()</code></td>
<td>Get the class FourCC code</td>
</tr>
</tbody>
</table>
Static Public Member Functions

```c
static void DumpRefCountingLeaks ()
{
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
```
### Static Protected Member Functions

<table>
<thead>
<tr>
<th>static Util::String</th>
<th>BuildSignature (const Util::Array&lt;CoreGraphics::VertexComponent &gt; &amp;c)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get sharing signature for a set of vertex components</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```
get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```
increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```
decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```
get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```
get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```
dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win360::Win360Barrier
Win360::Win360Barrier Class Reference

#include <win360barrier.h>
Detailed Description

Implements the 2 macros ReadWriteBarrier and MemoryBarrier.

ReadWriteBarrier prevents the compiler from re-ordering memory accesses across the barrier.

MemoryBarrier prevents the CPU from reordering memory access across the barrier (all memory access will be finished before the barrier is crossed).

(C) 2007 Radon Labs GmbH
Win360::Win360CalendarTime
Win360::Win360CalendarTime Class Reference

#include <win360calendartime.h>

Inheritance diagram for Win360::Win360CalendarTime:
Detailed Description

Win360 implementation of CalendarTime.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><strong>Month</strong></td>
</tr>
<tr>
<td></td>
<td><code>months enum</code></td>
</tr>
<tr>
<td>enum</td>
<td><strong>Weekday</strong></td>
</tr>
<tr>
<td></td>
<td><code>weekdays enum</code></td>
</tr>
<tr>
<td>typedef</td>
<td><code>unsigned int</code></td>
</tr>
<tr>
<td></td>
<td><strong>Year</strong></td>
</tr>
<tr>
<td></td>
<td><code>typedefs</code></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetYear (Year y)</code></td>
<td>set the year</td>
</tr>
<tr>
<td><code>GetYear ()</code></td>
<td>get the year</td>
</tr>
<tr>
<td><code>SetMonth (Month m)</code></td>
<td>set the month</td>
</tr>
<tr>
<td><code>GetMonth ()</code></td>
<td>get the month</td>
</tr>
<tr>
<td><code>SetWeekday (Weekday wd)</code></td>
<td>set the day-of-week</td>
</tr>
<tr>
<td><code>GetWeekday ()</code></td>
<td>get the day-of-week</td>
</tr>
<tr>
<td><code>SetDay (Day d)</code></td>
<td>set the day (of month)</td>
</tr>
<tr>
<td><code>GetDay ()</code></td>
<td>get the day (of month)</td>
</tr>
<tr>
<td><code>SetHour (Hour h)</code></td>
<td>set hour-of-day</td>
</tr>
<tr>
<td><code>GetHour ()</code></td>
<td>get hour-of-day</td>
</tr>
<tr>
<td><code>SetMinute (Minute m)</code></td>
<td>set minute-of-hour</td>
</tr>
<tr>
<td><code>GetMinute ()</code></td>
<td>get minute-of-hour</td>
</tr>
<tr>
<td><code>SetSecond (Second s)</code></td>
<td>set second-of-minute</td>
</tr>
<tr>
<td><code>GetSecond ()</code></td>
<td>get second-of-minute</td>
</tr>
<tr>
<td><code>SetMilliSecond (MilliSecond ms)</code></td>
<td>set milliseconds</td>
</tr>
<tr>
<td><code>GetMilliSecond ()</code></td>
<td>get milliseconds</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Timing::CalendarTime</td>
<td><strong>GetSystemTime</strong> ()</td>
<td>get the current system time</td>
</tr>
<tr>
<td>static Timing::CalendarTime</td>
<td><strong>GetLocalTime</strong> ()</td>
<td>get the current local time</td>
</tr>
<tr>
<td>static IO::FileTime</td>
<td><strong>SystemTimeToFileTime</strong> (const Timing::CalendarTime &amp;systemTime)</td>
<td>convert system time to file time</td>
</tr>
<tr>
<td>static Timing::CalendarTime</td>
<td><strong>FileTimeToSystemTime</strong> (const IO::FileTime &amp;fileTime)</td>
<td>convert file time to system time</td>
</tr>
<tr>
<td>static IO::FileTime</td>
<td><strong>LocalTimeToFileTime</strong> (const Timing::CalendarTime &amp;localTime)</td>
<td>convert local time to file time</td>
</tr>
<tr>
<td>static Timing::CalendarTime</td>
<td><strong>FileTimeToLocalTime</strong> (const IO::FileTime &amp;fileTime)</td>
<td>convert file time to local time</td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>Format</strong> (const Util::String &amp;fmtString, const Timing::CalendarTime &amp;calTime)</td>
<td>format to string</td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>MonthToString</strong> (Month m)</td>
<td>convert month to string</td>
</tr>
<tr>
<td>static Month</td>
<td><strong>StringToMonth</strong> (const Util::String &amp;str)</td>
<td>convert string to month</td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>WeekdayToString</strong> (Weekday d)</td>
<td>convert weekday to string</td>
</tr>
<tr>
<td>static Weekday</td>
<td><strong>StringToWeekday</strong> (const Util::String &amp;str)</td>
<td>convert string to weekday</td>
</tr>
</tbody>
</table>
Member Function Documentation

**CalendarTime**
Win360::Win360CalendarTime::GetSystemTime( ) [static]

get the current system time

Obtains the current system time. This does not depend on the current time zone.

Reimplemented from **Base::CalendarTimeBase**.

**CalendarTime**
Win360::Win360CalendarTime::GetLocalTime( ) [static]

get the current local time

Obtains the current local time (with time-zone adjustment).

Reimplemented from **Base::CalendarTimeBase**.

**String**
Base::CalendarTimeBase::Format( const Util::String & fmtString,
const Timing::CalendarTime calTime & ) [static, inherited]

format to string

Formats a calendar time into a string using the following substitution string:

{YEAR} - the year member
{MONTH} - the month member
{WEEKDAY} - the weekday member
{DAY} - the numerical day-in-month member
{HOUR} - the hour member
{MINUTE} - the minute member
{SECOND} - the second member
{MILLISECOND} - the millisecond member
Win360::Win360CriticalSection
Win360::Win360CriticalSection Class Reference

#include <win360criticalsection.h>

Inheritance diagram for Win360::Win360CriticalSection:

```plaintext
Win360::Win360CriticalSection
  <-
Threading::CriticalSection
```
Detailed Description

Win32/Xbox360-implementation of critical section. Critical section objects are used to protect a portion of code from parallel execution. Define a static critical section object and use its `Enter()` and `Leave()` methods to protect critical sections of your code.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Win360CriticalSection()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~Win360CriticalSection()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void Enter () const</code></td>
<td>enter the critical section</td>
</tr>
<tr>
<td><code>void Leave () const</code></td>
<td>leave the critical section</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win360::Win360Event
Win360::Win360Event Class Reference

#include <win360event.h>

Inheritance diagram for Win360::Win360Event:
Detailed Description

Win32/Xbox360 implementation of an event synchronization object.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Win360Event</strong></td>
<td>(bool manualReset=false)</td>
</tr>
<tr>
<td><strong>constructor</strong></td>
<td></td>
</tr>
<tr>
<td><strong>~Win360Event</strong></td>
<td>()</td>
</tr>
<tr>
<td><strong>void Signal</strong></td>
<td>()</td>
</tr>
<tr>
<td><strong>void Reset</strong></td>
<td>()</td>
</tr>
<tr>
<td><strong>void Wait</strong></td>
<td>() const</td>
</tr>
<tr>
<td><strong>bool WaitTimeout</strong></td>
<td>(int ms) const</td>
</tr>
<tr>
<td><strong>bool Peek</strong></td>
<td>() const</td>
</tr>
</tbody>
</table>

- **Win360Event**: Constructor
- **~Win360Event**: Destructor
- **Signal**: Signal the event
- **Reset**: Reset the event (only if manual reset)
- **Wait**: Wait for the event to become signalled
- **WaitTimeout**: Wait for the event with timeout in millisecs
- **Peek**: Check if event is signalled
Member Function Documentation

`bool Win360::Win360Event::WaitTimeout (int timeoutInMilliSec ) const [inline]`

wait for the event with timeout in millisecs

Waits for the event to become signaled with a specified timeout in milliseconds. If the method times out it will return false, if the event becomes signalled within the timeout it will return true.

`bool Win360::Win360Event::Peek ( ) const [inline]`

check if event is signalled

This checks if the event is signalled and returns immediately.
Win360::Win360FileTime
Win360::Win360FileTime Class Reference

#include <win360filetime.h>
Detailed Description

Implements a Win32/Xbox360-specific file-access time stamp.

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Win360FileTime ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>Win360FileTime (const Util::String &amp;str)</code></td>
<td>Construct from string</td>
</tr>
<tr>
<td><code>Util::String AsString () const</code></td>
<td>Convert to string</td>
</tr>
</tbody>
</table>
### Friends

<table>
<thead>
<tr>
<th>bool</th>
<th><code>operator==(const Win360FileTime &amp;a, const Win360FileTime &amp;b)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>operator ==</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>operator!=(const Win360FileTime &amp;a, const Win360FileTime &amp;b)</code></td>
</tr>
<tr>
<td></td>
<td><code>operator !=</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>operator&gt;(const Win360FileTime &amp;a, const Win360FileTime &amp;b)</code></td>
</tr>
<tr>
<td></td>
<td><code>operator &gt;</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>operator&lt;(const Win360FileTime &amp;a, const Win360FileTime &amp;b)</code></td>
</tr>
<tr>
<td></td>
<td><code>operator &lt;</code></td>
</tr>
</tbody>
</table>
Win360::Win360FSWrapper
Win360::Win360FSWrapper Class Reference

#include <win360fswrapper.h>

Inheritance diagram for Win360::Win360FSWrapper:

Win360::Win360FSWrapper

IO::FSShared
Detailed Description

Internal filesystem wrapper for Win32/Xbox360. All paths must be native paths (i.e. not contain Nebula assigns).

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OpenFile</strong></td>
<td>(const <em>Util::String</em> &amp;path, <em>IO::Stream::AccessMode</em> accessMode, <em>IO::Stream::AccessPattern</em> accessPattern, DWORD flagsAndAttributes=0) open a file</td>
</tr>
<tr>
<td><strong>CloseFile</strong></td>
<td>(Handle h) close a file</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>(Handle h, const void *buf, <em>IO::Stream::Size</em> numBytes) write to a file</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>(Handle h, void *buf, <em>IO::Stream::Size</em> numBytes) read from a file</td>
</tr>
<tr>
<td><strong>Seek</strong></td>
<td>(Handle h, <em>IO::Stream::Offset</em> offset, <em>IO::Stream::SeekOrigin</em> orig) seek in a file</td>
</tr>
<tr>
<td><strong>Tell</strong></td>
<td>(Handle h) get position in file</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>(Handle h) flush a file</td>
</tr>
<tr>
<td><strong>Eof</strong></td>
<td>(Handle h) return true if at end-of-file</td>
</tr>
<tr>
<td><strong>GetFileSize</strong></td>
<td>(Handle h) get size of a file in bytes</td>
</tr>
<tr>
<td><strong>SetReadOnly</strong></td>
<td>(const <em>Util::String</em> &amp;path, bool readOnly) set read-only status of a file</td>
</tr>
<tr>
<td><strong>IsReadOnly</strong></td>
<td>(const <em>Util::String</em> &amp;path) get read-only status of a file</td>
</tr>
<tr>
<td><strong>DeleteFile</strong></td>
<td>(const <em>Util::String</em> &amp;path) delete a file</td>
</tr>
<tr>
<td><strong>DeleteDirectory</strong></td>
<td>(const <em>Util::String</em> &amp;path) delete an empty directory</td>
</tr>
</tbody>
</table>
static bool FileExists (const Util::String &path)
return true if a file exists

static bool DirectoryExists (const Util::String &path)
return true if a directory exists

static void SetFileWriteTime (const Util::String &path, IO::FileTime fileTime)
set the write-access time stamp of a file

static IO::FileTime GetFileWriteTime (const Util::String &path)
get the last write-access time stamp of a file

static bool CreateDirectory (const Util::String &path)
create a directory

static Util::Array<Util::String> ListFiles (const Util::String &dirPath, const Util::String &pattern)
list all files in a directory

static Util::Array<Util::String> ListDirectories (const Util::String &dirPath, const Util::String &pattern)
list all subdirectories in a directory

static Util::String GetUserDirectory ()
get path to the current user's home directory (for user: standard assign)

static Util::String GetAppDataDirectory ()
get path to the current user's appdata directory (for appdata: standard assign)

static Util::String GetTempDirectory ()
get path to the current user's temp directory (for temp: standard assign)

static Util::String GetHomeDirectory ()
get path to the current application directory (for home: standard assign)

static Util::String GetBinDirectory ()
get path to the current bin directory (for bin: standard assign)

static Util::String GetProgramsDirectory ()
get path to the "c:/program files" directory

static bool IsDeviceName (const Util::String &str)
return true when the string is a device name (e.g. "C:"
Member Function Documentation

Win360FSWrapper::Handle
Win360::Win360FSWrapper::OpenFile
( const Util::String & path,
  IO::Stream::AccessMode accessMode,
  IO::Stream::AccessPattern accessPattern,
  DWORD flagsAndAttributes = 0 )
[static]

open a file

Open a file using the Xbox360 function CreateFile(). Returns a handle to the file which must be passed to the other Win360FSWrapper file methods. If opening the file fails, the function will return 0. The filename must be a native Xbox360 path (no assigns, etc...).

void
Win360::Win360FSWrapper::CloseFile( Handle handle ) [static]

close a file

Closes a file opened by Win360FSWrapper::OpenFile().

Stream::Position
Win360::Win360FSWrapper::Tell( Handle handle ) [static]

get position in file

Get current position in file.

void
Win360::Win360FSWrapper::Flush( Handle handle ) [static]

flush a file

Flush unwritten data to file.

bool
Win360::Win360FSWrapper::Eof( Handle handle ) [static]
return true if at end-of-file

Returns true if current position is at end of file.

Stream::Size
Win360::Win360FSWrapper::GetFileSize( Handle handle ) [static]

get size of a file in bytes

Returns the size of a file in bytes.

void
Win360::Win360FSWrapper::SetReadOnly( const Util::String &path,
                                      bool readOnly ) [static]

set read-only status of a file

Set the read-only status of a file. This method does nothing on the Xbox360.

bool
Win360::Win360FSWrapper::IsReadOnly( const Util::String &path ) [static]

get read-only status of a file

Get the read-only status of a file. This method always returns true on the Xbox360.

bool
Win360::Win360FSWrapper::DeleteFile( const Util::String &path ) [static]

delete a file

Deletes a file. Returns true if the operation was successful. The delete will fail if the file doesn't exist or the file is read-only.

bool
Win360::Win360FSWrapper::DeleteDirectory( const Util::String &path ) [static]
delete an empty directory

Delete an empty directory. Returns true if the operation was successful.

```cpp
bool Win360::Win360FSWrapper::FileExists(const Util::String path) [static]
```

return true if a file exists

Return true if a file exists.

```cpp
bool Win360::Win360FSWrapper::DirectoryExists(const Util::String path) [static]
```

return true if a directory exists

Return true if a directory exists.

```cpp
void Win360::Win360FSWrapper::SetFileWriteTime(const Util::String path, IO::FileTime fileTime) [static]
```

set the write-access time stamp of a file

Set the write-access time stamp of a file.

```cpp
FileTime Win360::Win360FSWrapper::GetFileWriteTime(const Util::String path) [static]
```

get the last write-access time stamp of a file

Return the last write-access time to a file.

```cpp
bool Win360::Win360FSWrapper::CreateDirectory(const Util::String path) [static]
```
create a directory

Creates a new directory.

```
Array< String >
Win360::Win360FSWrapper::ListFiles ( const Util::String dirPath, &
const Util::String pattern & )

[static]
```

list all files in a directory

Lists all files in a directory, filtered by a pattern.

```
Array< String >
Win360::Win360FSWrapper::ListDirectories ( const Util::String dirPath, &
const Util::String pattern & )

[static]
```

list all subdirectories in a directory

Lists all subdirectories in a directory, filtered by a pattern. This will not return the special directories ".." and ".".

```
String
Win360::Win360FSWrapper::GetUserDirectory ( )

[static]
```

get path to the current user's home directory (for user: standard assign)

NOTE: The user: standard assign is not supported on the 360.

```
String
Win360::Win360FSWrapper::GetAppDataDirectory ( )

[static]
```

get path to the current user's appdata directory (for appdata: standard assign)
NOTE: The appdata: standard assign is not supported on the 360.

```c
String Win360::Win360FSWrapper::GetTempDirectory() [static]
```

get path to the current user's temp directory (for temp: standard assign)

NOTE: The temp standard assign is not supported on the 360 (only on Devkits!)

```c
String Win360::Win360FSWrapper::GetHomeDirectory() [static]
```

get path to the current application directory (for home: standard assign)

This method should return the installation directory of the application.

```c
String Win360::Win360FSWrapper::GetBinDirectory() [static]
```

get path to the current bin directory (for bin: standard assign)

This method should return the directory where the application executable is located.

```c
String Win360::Win360FSWrapper::GetProgramsDirectory() [static]
```

get path to the "c:/program files" directory

NOTE: The programs: standard assign is not supported on the 360.

```c
bool Win360::Win360FSWrapper::IsDeviceName( const Util::String str ) [static]
```

return true when the string is a device name (e.g. "C:\")

Return true if the provided string is a device name.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win360::Win360Heap
Win360::Win360Heap Class Reference

#include <win360heap.h>
Detailed Description

Win32/Xbox360 implementation of the class `Memory::Heap`. Under Win32, the LowFragmentationHeap feature is generally turned on.

(C) 2006 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Win360Heap</strong></td>
<td>(const char *name, size_t initialSize=0, size_t maxSize=0)</td>
</tr>
<tr>
<td></td>
<td>constructor (name must be static string!)</td>
</tr>
<tr>
<td><strong>~Win360Heap</strong></td>
<td>()</td>
</tr>
<tr>
<td></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>GetName</strong></td>
<td>() const</td>
</tr>
<tr>
<td></td>
<td>get heap name</td>
</tr>
<tr>
<td><strong>Alloc</strong></td>
<td>(size_t size)</td>
</tr>
<tr>
<td></td>
<td>allocate a block of memory from the heap</td>
</tr>
<tr>
<td><strong>Realloc</strong></td>
<td>(void *ptr, size_t newSize)</td>
</tr>
<tr>
<td></td>
<td>re-allocate a block of memory</td>
</tr>
<tr>
<td><strong>Free</strong></td>
<td>(void *ptr)</td>
</tr>
<tr>
<td></td>
<td>free a block of memory which has been allocated from this heap</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

```plaintext
static void Setup ()

static setup method (called by Core::SysFunc::Setup)
```
Member Function Documentation

void Win360::Win360Heap::Setup() [static]

static setup method (called by Core::SysFunc::Setup)

This method must be called at the beginning of the application before any threads are spawned.
Win360::Win360Interlocked
Win360::Win360Interlocked Class Reference

#include <win360interlocked.h>

Inheritance diagram for Win360::Win360Interlocked:

```
Win360::Win360Interlocked

Threading::Interlocked
```


Detailed Description

Provides simple atomic operations on shared variables.

(C) 2006 Radon Labs GmbH
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Increment</code></td>
<td><code>interlocked increment, return result</code></td>
</tr>
<tr>
<td><code>Decrement</code></td>
<td><code>interlocked decrement, return result</code></td>
</tr>
<tr>
<td><code>Add</code></td>
<td><code>interlocked add</code></td>
</tr>
<tr>
<td><code>Exchange</code></td>
<td><code>interlocked exchange</code></td>
</tr>
<tr>
<td><code>CompareExchange</code></td>
<td><code>interlocked compare-exchange</code></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](https://www.doxygen.nl) at Fri Mar 26 15:21:51 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win360::Win360IpAddress
Win360::Win360IpAddress Class Reference

#include <win360ipaddress.h>
Detailed Description

NOTE: Socket network communication on the Xbox360 is only provided for debugging and development purposes. For actual multiplayer and Xbox Live related stuff, use the Xbox-specific add-on modules! (which don't exist yet, ha).

Represents an IP address, consisting of a IPv4 host address and a port number. Performs automatic name lookup on the host name. Can extract address information from an URI and automatically converts host names to addresses, and offers the special hostnames "localhost", "any", "broadcast", "self" and "inetself" where:

- "localhost" will translate to 127.0.0.1
- "any" will translate to INADDR_ANY, which is 0.0.0.0
- "broadcast" will translate to INADDR_BROADCAST, which is 255.255.255.255
- "self" will translate to the first valid tcp/ip address for this host (there may be more then one address bound to the host)
- "inetself" will translate to the first host address which is not a LAN address (which is not a class A, B, or C network) if none such exists the address will fall back to "self"

(C) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Win360IpAddress()</code></td>
<td>Default constructor</td>
</tr>
<tr>
<td><code>Win360IpAddress(const Win360IpAddress &amp;rhs)</code></td>
<td>Copy constructor</td>
</tr>
<tr>
<td><code>Win360IpAddress(const IO::URI &amp;uri)</code></td>
<td>Construct from URI</td>
</tr>
<tr>
<td><code>Win360IpAddress(const Util::String &amp;hostName, ushort portNumber)</code></td>
<td>Construct from host name and port number</td>
</tr>
<tr>
<td>bool <code>operator==(const Win360IpAddress &amp;rhs)</code> const</td>
<td>Equality operator</td>
</tr>
<tr>
<td>bool <code>operator&lt;(const Win360IpAddress &amp;rhs)</code> const</td>
<td>Less-than operator</td>
</tr>
<tr>
<td>bool <code>operator&gt;(const Win360IpAddress &amp;rhs)</code> const</td>
<td>Greater-than operator</td>
</tr>
<tr>
<td>void <code>ExtractFromUri(const IO::URI &amp;uri)</code></td>
<td>Extract host name and port number from URI</td>
</tr>
<tr>
<td>void <code>SetHostName(const Util::String &amp;hostName)</code></td>
<td>Set host name</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetHostName()</code> const</td>
<td>Get host name</td>
</tr>
<tr>
<td>void <code>SetPort(ushort port)</code></td>
<td>Set port number</td>
</tr>
<tr>
<td>ushort <code>GetPort()</code> const</td>
<td>Get port number</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetHostAddr()</code> const</td>
<td>Get the ip address resulting from the host name as string</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Win360::Win360IpAddress::ExtractFromUri ( IO::URI uri )

extract host name and port number from URI

Extract the host name and optionally the port number from the
provided URI. If no port number is set in the URI, the current port
number will be left as is. If the host name is empty, it will be set to
"localhost".

void Win360::Win360IpAddress::SetHostName ( Util::String hostName )

set host name

Set the host name, and immediately convert it to an ip address. This
accepts the special hostnames "any", "broadcast", "localhost", "self"
and "inetself". The result ip address can be returned in string form with
the method GetAddrAsString().

const String & Win360::Win360IpAddress::GetHostName ( ) const

get host name

Get the host name.

void Win360::Win360IpAddress::SetPort ( ushort port )

set port number

Set the port number. Will be translated to network byte order internally.

ushort Win360::Win360IpAddress::GetPort ( ) const
get port number

Get the port number in host byte order.

```
const String &
Win360::Win360IpAddress::GetHostAddr()
```

get the ip address resulting from the host name as string

Return the in address as string.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win360::Win360MemoryPool
Win360::Win360MemoryPool Class Reference

#include <win360memorypool.h>
Detailed Description

A simple thread-safe memory pool. Memory pool items are 16-byte aligned.

FIXME:

- debug: overwrite memory blocks with pattern
- debug: check for double-free
- debug: check for mem-leaks
- debug: list memory pools in Debug HTML page!

(C) 2009 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Win360MemoryPool ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~Win360MemoryPool ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>Setup (Memory::HeapType heapType, uint blockSize, uint numBlocks)</strong></td>
<td>setup the memory pool</td>
</tr>
<tr>
<td>void <strong>Alloc ()</strong></td>
<td>allocate a block from the pool (NOTE: returns 0 if pool exhausted!)</td>
</tr>
<tr>
<td>void *<em>Free (void <em>ptr)</em></em></td>
<td>deallocate a block from the pool</td>
</tr>
<tr>
<td>bool *<em>IsPoolBlock (void <em>ptr) const</em></em></td>
<td>return true if block is owned by this pool</td>
</tr>
<tr>
<td>uint <strong>GetNumBlocks () const</strong></td>
<td>get number of allocated blocks in pool</td>
</tr>
<tr>
<td>uint <strong>GetBlockSize () const</strong></td>
<td>get block size</td>
</tr>
<tr>
<td>uint <strong>GetAlignedBlockSize () const</strong></td>
<td>get aligned block size</td>
</tr>
<tr>
<td>uint <strong>GetPoolSize () const</strong></td>
<td>get pool size</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static uint</th>
<th><strong>ComputeAlignedBlockSize</strong> (uint blockSize)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>compute the actual block size including alignment and management data</td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Win360::Win360MemoryPool::Setup (Memory::HeapType heapType_,
uint blockSize_,
uint numBlocks_)

setup the memory pool

NOTE: name must be a static string!
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win360::Win360Socket
Win360::Win360Socket Class Reference

#include <win360socket.h>

Inheritance diagram for Win360::Win360Socket:

```
Core::RefCounted

Win360::Win360Socket

Net::Socket
```
Detailed Description

NOTE: Socket network communication on the Xbox360 is only provided for debugging and development purposes. For actual multiplayer and Xbox Live related stuff, use the Xbox-specific add-on modules!

A lowlevel socket wrapper class. This class is primarily a platform abstraction class, not an "end-user-class". Usually a Nebula3 application doesn't use this class directly but instead uses the higher level classes like TcpServer, TclClient, etc...

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>protocol types</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th>ErrorCode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>error codes</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Win360Socket ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~Win360Socket ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>bool <strong>Open (Protocol p)</strong></td>
<td>open the socket</td>
</tr>
<tr>
<td>void <strong>Close ()</strong></td>
<td>close the socket</td>
</tr>
<tr>
<td>bool <strong>IsOpen ()</strong> const</td>
<td>return true if the socket is open</td>
</tr>
<tr>
<td>ErrorCode <strong>GetErrorCode ()</strong></td>
<td>get the last error code</td>
</tr>
<tr>
<td>Util::String <strong>getErrorString ()</strong></td>
<td>get the last error string</td>
</tr>
<tr>
<td>void <strong>SetAddress (const Net::IpAddress &amp;a)</strong></td>
<td>set internet address of socket</td>
</tr>
<tr>
<td>const Net::IpAddress &amp; <strong>GetAddress ()</strong> const</td>
<td>get internet address of socket</td>
</tr>
<tr>
<td>void <strong>SetBroadcast (bool b)</strong></td>
<td>set the broadcast flag (SO_BROADCAST)</td>
</tr>
<tr>
<td>bool <strong>GetBroadcast ()</strong></td>
<td>get the broadcast flag</td>
</tr>
<tr>
<td>void <strong>SetKeepAlive (bool b)</strong></td>
<td>set the keepalive flag (SO_KEEPALIVE)</td>
</tr>
<tr>
<td>bool <strong>GetKeepAlive ()</strong></td>
<td>get the keepalive flag</td>
</tr>
<tr>
<td>void <strong>SetReUseAddr (bool b)</strong></td>
<td>set reuseaddr flag (SO_REUSEADDR)</td>
</tr>
<tr>
<td>bool <strong>GetReUseAddr ()</strong></td>
<td>get reuseaddr flag</td>
</tr>
<tr>
<td>void <strong>SetNoDelay (bool b)</strong></td>
<td>set nodelay flag (TCP_NODELAY)</td>
</tr>
<tr>
<td>bool <strong>GetNoDelay ()</strong></td>
<td>get nodelay flag</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>get nodelay flag</td>
<td>set receive buffer size</td>
</tr>
<tr>
<td><code>void SetRecvBufSize (SizeT s)</code></td>
<td>set receive buffer size</td>
</tr>
<tr>
<td><code>SizeT GetRecvBufSize ()</code></td>
<td>get receive buffer size</td>
</tr>
<tr>
<td><code>void SetSendBufSize (SizeT s)</code></td>
<td>set send buffer size</td>
</tr>
<tr>
<td><code>SizeT GetSendBufSize ()</code></td>
<td>get send buffer size</td>
</tr>
<tr>
<td><code>void SetBlocking (bool b)</code></td>
<td>set blocking mode (FIONBIO)</td>
</tr>
<tr>
<td><code>bool GetBlocking () const</code></td>
<td>get blocking mode</td>
</tr>
<tr>
<td><code>SizeT GetMaxMsgSize ()</code></td>
<td>get the maximum message size that can be sent atomically</td>
</tr>
<tr>
<td><code>bool Bind ()</code></td>
<td>bind socket to ip address</td>
</tr>
<tr>
<td><code>bool IsBound () const</code></td>
<td>return true if the socket is bound to an address</td>
</tr>
<tr>
<td><code>bool Listen ()</code></td>
<td>listen for incoming connections (for server sockets)</td>
</tr>
<tr>
<td><code>bool Accept (Ptr&lt; Net::Socket &gt; &amp;outSocket)</code></td>
<td>accept incoming connection, return a new socket (for server sockets)</td>
</tr>
<tr>
<td><code>Result Connect ()</code></td>
<td>connect to the sockets address (for client sockets)</td>
</tr>
<tr>
<td><code>bool IsConnected ()</code></td>
<td>test if the socket is currently connected</td>
</tr>
<tr>
<td><code>Result Send (const void *buf, SizeT numBytes, SizeT &amp;bytesSent)</code></td>
<td>send raw data into the socket</td>
</tr>
<tr>
<td><code>bool HasRecvData ()</code></td>
<td>return true if recv data is available at the socket</td>
</tr>
<tr>
<td><code>Result Recv (void *buf, SizeT bufSize, SizeT &amp;bytesReceived)</code></td>
<td>receive raw data from the socket</td>
</tr>
<tr>
<td><code>SendTo (const void *buf, SizeT numBytes, uint</code></td>
<td>send raw data into the socket</td>
</tr>
<tr>
<td>Result</td>
<td>addr, ushort port, SizeT &amp;bytesSent</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Result</td>
<td><code>RecvFrom (void *buf, SizeT bufSize, uint addr, ushort port, SizeT &amp;bytesReceived)</code></td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</code></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>bool IsA (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetClassName () const</code></td>
<td>get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC GetClassFourCC () const</code></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><code>InitNetwork()</code></td>
<td>static initializer method (called by SysFunc::Setup())</td>
</tr>
<tr>
<td>static bool</td>
<td><code>IsNetworkInitialized()</code></td>
<td>is network initialized</td>
</tr>
<tr>
<td>static void</td>
<td><code>DumpRefCountingLeaks()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Win360::Win360Socket::InitNetwork() [static]

static initializer method (called by SysFunc::Setup())

This is a one-time init for the Windows Sockets system. The method is called from SysFunc::Setup() once at startup before any threads are launched.

void Win360::Win360Socket::SetAddress(const Net::IpAddress& a) [inline]

set internet address of socket

Set internet address of socket.

const Net::IpAddress& Win360::Win360Socket::GetAddress() const [inline]

get internet address of socket

Get internet address of socket.

void Win360::Win360Socket::SetBlocking(bool b)

set blocking mode (FIONBIO)

Set the socket to blocking mode.

bool Win360::Win360Socket::Bind()

bind socket to ip address

Bind the socket to its ip address set with SetAddress() and SetPort(). After binding the socket to an address, call the Listen() method to wait for incoming connections. This method only makes sense for server
sockets.

```cpp
bool Win360::Win360Socket::Listen() {
  // Code to listen for incoming connections (for server sockets)
  // Wait for incoming connections to a server socket. Call this method on
  // server side after binding the socket to its address.
}
```

```cpp
bool Win360::Win360Socket::Accept(Ptr<Net::Socket> &outSocket) {
  // Code to accept incoming connection, return a new socket (for server sockets)
  // Accept an incoming connection to a server socket. This will spawn a
  // new socket for the connection which will be returned in the provided
  // pointer reference. The address of the returned socket will be set to the
  // address of the "connecting entity".
}
```

```cpp
Win360Socket::Result Win360::Win360Socket::Connect() {
  // Code to connect to the sockets address (for client sockets)
  // Connect to a server socket. This method is called by a client socket to
  // connect to a server socket identified by the socket object's address. A
  // non-blocking socket will return immediately with WouldBlock, since the
  // connection cannot be established immediately. In this case, just
  // continue to call Connect() until the method returns Success, or
  // alternative, check the IsConnected() method, which will also return
  // true once the connection has been establish.
}
```

```cpp
bool Win360::Win360Socket::IsConnected() {
  // Code to test if the socket is currently connected
  // This tests if the socket is actually connected by doing a select() on the
  // socket to probe for writability. So the IsConnected() method basically
  // checks whether data can be sent through the socket.
}
```
Win360Socket::Result
Win360::Win360Socket::Send(const void *buf,
    SizeT numBytes,
    SizeT &bytesSent
)

send raw data into the socket

Send raw data into the socket. Note that depending on the buffer size of the underlying socket implementation and other sockets, the method may not be able to send all provided data. In this case, the returned content of bytesSent will be less than numBytes, even though the return value will be Success. It is up to the caller to handle the extra data which hasn't been sent with the current call.

bool
Win360::Win360Socket::HasRecvData()

return true if recv data is available at the socket

This method checks if the socket has received data available. Use this method in a loop with Recv() to get all data waiting at the socket. This method will never block.

Win360Socket::Result
Win360::Win360Socket::Recv(void *buf,
    SizeT bufSize,
    SizeT &bytesReceived
)

receive raw data from the socket

Receive raw data from a socket and write the received data into the provided buffer. On a blocking socket this method will block until data arrives at the socket. A non-blocking socket would immediately return in this case with a WouldBlock result. When valid data has been received the method will return with a Success result and the bytesReceived argument will contain the number of received bytes. It is not guaranteed that a single receive will return all data waiting on
the socket. To make sure that the socket is really empty, call \texttt{Recv()} in a loop until \texttt{HasRecvData()} returns false. When the socket has been gracefully closed by the other side, the method will return with a Closed return value. Everything else will return with an Error return code. Call \texttt{GetErrorCode()} or \texttt{GetErrorString()} to find out more in this case.

```
Win360Socket::Result Win360::Win360Socket::SendTo(const void * buf, 
        SizeT numBytes, 
        uint addr, 
        ushort port, 
        SizeT & bytesSent)
```

send raw data to address for connectionless sockets

\textbf{FIXME}: this is the send method for connectionless sockets using the UDP protocol.

```
Win360Socket::Result Win360::Win360Socket::RecvFrom(void * buf, 
        SizeT bufSize, 
        uint addr, 
        ushort port, 
        SizeT & bytesReceived)
```

receive raw data from address for connectionless sockets

\textbf{FIXME}: this is the recv method for connectionless socket using the UDP protocol.

```
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.
void Core::RefCounted::AddRef( ) [inline, inherited]
increment refcount by one
Increment the refcount of the object.

void Core::RefCounted::Release( ) [inline, inherited]
decrement refcount and destroy object if refcount is zero
Decrement the refcount and destroy object if refcount is zero.

const Util::String & Core::RefCounted::GetClassName( ) const [inline, inherited]
get the class name
Get the class name of the object.

Util::FourCC Core::RefCounted::GetClassFourCC( ) const [inline, inherited]
get the class FourCC code
Get the class FourCC of the object.

void Core::RefCounted::DumpRefCountingLeaks( ) [static, inherited]
dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win360::Win360Thread
Win360::Win360Thread Class Reference

#include <win360thread.h>

Inheritance diagram for Win360::Win360Thread:
Detailed Description

Win32/Xbox360 implementation of thread class.

(C) 2006 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>thread priorities</td>
</tr>
</tbody>
</table>
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win360Thread ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~Win360Thread ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void SetPriority (Priority p)</td>
<td>set the thread priority</td>
</tr>
<tr>
<td>Priority GetPriority () const</td>
<td>get the thread priority</td>
</tr>
<tr>
<td>void SetCoreId (System::Cpu::CoreId coreId)</td>
<td>set cpu core on which the thread should be running</td>
</tr>
<tr>
<td>System::Cpu::CoreId GetCoreId () const</td>
<td>get the cpu core on which the thread should be running</td>
</tr>
<tr>
<td>void SetStackSize (SizeT s)</td>
<td>set stack size in bytes (default is 4 KByte)</td>
</tr>
<tr>
<td>SizeT GetStackSize () const</td>
<td>get stack size</td>
</tr>
<tr>
<td>void SetName (const Util::String &amp;n)</td>
<td>set thread name</td>
</tr>
<tr>
<td>const Util::String &amp; GetName () const</td>
<td>get thread name</td>
</tr>
<tr>
<td>void Start ()</td>
<td>start executing the thread code, returns when thread has actually started</td>
</tr>
<tr>
<td>void Stop ()</td>
<td>request threading code to stop, returns when thread has actually finished</td>
</tr>
<tr>
<td>bool IsRunning () const</td>
<td>return true if thread has been started</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::String &amp;className) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::String &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by string</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the class FourCC code</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><strong>YieldThread</strong> ()</td>
<td>yield the thread (gives up current time slice)</td>
</tr>
<tr>
<td>static void</td>
<td><strong>SetMyThreadName</strong> (const char *n)</td>
<td>set thread name from within thread context</td>
</tr>
<tr>
<td>static const char *</td>
<td><strong>GetMyThreadName</strong> ()</td>
<td>obtain name of thread from within thread context</td>
</tr>
<tr>
<td>static Threading::ThreadId</td>
<td><strong>GetMyThreadId</strong> ()</td>
<td>get the thread ID of this thread</td>
</tr>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void</td>
<td><code>EmitWakeupSignal</code> ()</td>
<td><em>override this method if your thread loop needs a wakeup call before stopping</em></td>
</tr>
<tr>
<td>virtual void</td>
<td><code>DoWork</code> ()</td>
<td><em>this method runs in the thread context</em></td>
</tr>
<tr>
<td>bool</td>
<td><code>ThreadStopRequested</code> () const</td>
<td><em>check if stop is requested, call from <code>DoWork()</code> to see if the thread proc should quit</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
template <typename T>
void Win360::Win360Thread::SetName(T* const n) [inline]
```

set thread name

Set the thread's name. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

```cpp
const Util::String & Win360::Win360Thread::GetName() const [inline]
```

get thread name

Get the thread's name. This is the vanilla method which returns the name member. To obtain the current thread's name from anywhere in the thread's execution context, call the static method `Thread::GetMyThreadName()`.

```cpp
void Win360::Win360Thread::Start() [inline]
```

start executing the thread code, returns when thread has actually started

Start the thread, this creates a Win32 thread and calls the static ThreadProc, which in turn calls the virtual `DoWork()` class of this object. The method waits for the thread to start and then returns.

```cpp
void Win360::Win360Thread::Stop() [inline]
```

request threading code to stop, returns when thread has actually finished

This stops the thread by signalling the stopRequestEvent and waits for the thread to actually quit. If the thread code runs in a loop it should
use the IsStopRequested() method to see if the thread object wants it to shutdown. If so DoWork() should simply return.

Reimplemented in Jobs::TPWorkerThread.

```cpp
bool Win360::Win360Thread::IsRunning() const
return true if thread has been started

Returns true if the thread is currently running.
```

```cpp
void Win360::Win360Thread::YieldThread() [static]
yield the thread (gives up current time slice)

The yield function is empty on Win32 and Xbox360.
```

```cpp
void Win360::Win360Thread::SetMyThreadName(const char * n ) [static]
set thread name from within thread context

Static method which sets the name of this thread. This is called from within ThreadProc. The string pointed to must remain valid until the thread is terminated!
```

```cpp
const char * Win360::Win360Thread::GetMyThreadName() [static]
obtain name of thread from within thread context

Static method to obtain the current thread name from anywhere in the thread's code.
```

```cpp
Threading::ThreadId Win360::Win360Thread::GetMyThreadId() [static]
get the thread ID of this thread
Static method which returns the ThreadId of this thread.

```cpp
void Win360::Win360Thread::EmitWakeupSignal() [protected, virtual]
```

override this method if your thread loop needs a wakeup call before stopping

This method is called by `Thread::Stop()` after setting the stopRequest event and before waiting for the thread to stop. If your thread runs a loop and waits for jobs it may need an extra wakeup signal to stop waiting and check for the `ThreadStopRequested()` event. In this case, override this method and signal your event object.

Reimplemented in `Jobs::TPWorkerThread`, and `Messaging::BlockingHandlerThread`.

```cpp
void Win360::Win360Thread::DoWork() [protected, virtual]
```

this method runs in the thread context

This method should be derived in a Thread subclass and contains the actual code which is run in the thread. The method must not call C-Lib functions under `Win32`. To terminate the thread, just return from this function. If `DoWork()` runs in an infinite loop, call `ThreadStopRequested()` to check whether the Thread object wants the thread code to quit.

Reimplemented in `FrameSync::FrameSyncHandlerThread`, `Jobs::TPWorkerThread`, `Messaging::BlockingHandlerThread`, and `Messaging::RunThroughHandlerThread`.

```cpp
bool Win360::Win360Thread::ThreadStopRequested() const [inline, protected]
```

check if stop is requested, call from `DoWork()` to see if the thread proc should quit

If the derived `DoWork()` method is running in a loop it must regularly check if the process wants the thread to terminate by calling
ThreadStopRequested() and simply return if the result is true. This will cause the thread to shut down.

```cpp
int Core::RefCounted::GetRefCount() const [inline, inherited]
```

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef() [inline, inherited]
```

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release() [inline, inherited]
```

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String & Core::RefCounted::GetClassName() const [inline, inherited]
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]
```

get the class FourCC code

Get the class FourCC of the object.

```cpp
void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]
```

dump refcounting leaks, call at end of application (NEBULA3_DEBUG
builds only!

This method should be called as the very last before an application exits.
Win360::Win360ThreadBarrier
Win360::Win360ThreadBarrier Class Reference

#include <win360threadbarrier.h>

Inheritance diagram for Win360::Win360ThreadBarrier:
Detailed Description

Block until all threads have arrived at the barrier.

(C) 2009 Radon Labs GmbH
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Win360ThreadBarrier()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~Win360ThreadBarrier()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>void <code>Setup(SizeT numThreads)</code></td>
<td>setup the object with the number of threads</td>
</tr>
<tr>
<td>bool <code>IsValid()</code> const</td>
<td>return true if the object has been setup</td>
</tr>
<tr>
<td>bool <code>Arrive()</code></td>
<td>enter thread barrier, return false if not all threads have arrived yet</td>
</tr>
<tr>
<td>void <code>Wait()</code></td>
<td>call after <code>Arrive()</code> returns false to wait for other threads</td>
</tr>
<tr>
<td>void <code>SignalContinue()</code></td>
<td>call after <code>Arrive()</code> returns true to resume all threads</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool
Win360::Win360ThreadBarrier::Arrive() [inline]

enter thread barrier, return false if not all threads have arrived yet

Notify arrival at thread-sync point, return false if not all threads have arrived yet, and true if all threads have arrived. If the method returns false, you should immediately call Wait(), if the method returns true, the caller has a chance to perform some actions which should happen before threads continue, and then call the SignalContinue() method.

void
Win360::Win360ThreadBarrier::Wait() [inline]

call after Arrive() returns false to wait for other threads

This method should be called when Arrive() returns false. It will put the thread to sleep because not all threads have arrived yet. When the method returns, all threads have arrived at the sync point.

NOTE: sometimes both the render and the main thread arrive here with the outstandingThreads member set to 1 (from two) causing both thread to be waiting indefinitely.

void
Win360::Win360ThreadBarrier::SignalContinue() [inline]

call after Arrive() returns true to resume all threads

This method should be called after Arrive() returns true. This means that all threads have arrived at the sync point and execution of all threads may resume.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win360::Win360ThreadId
Win360::Win360ThreadId Class Reference

#include <win360threadid.h>
Detailed Description

A thread id uniquely identifies a thread within the process.

(C) 2008 Radon Labs GmbH
Win360::Win360Timer
Win360::Win360Timer Class Reference

#include <win360timer.h>

Inheritance diagram for Win360::Win360Timer:
Detailed Description

Win32/Xbox360 implementation of the Time::Timer class. Uses the QueryPerformanceCounter() functions.

Todo:
solve multiprocessor issues of QueryPerformanceCounter() (different processors may return different PerformanceFrequency values, thus, threads should be prevented from switching between processors with thread affinities).

(C) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Win360Timer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>void <strong>Start ()</strong></td>
<td>Start/continue the timer</td>
</tr>
<tr>
<td>void <strong>Stop ()</strong></td>
<td>Stop the timer</td>
</tr>
<tr>
<td>void <strong>Reset ()</strong></td>
<td>Reset the timer</td>
</tr>
<tr>
<td>bool <strong>Running () const</strong></td>
<td>return true if currently running</td>
</tr>
<tr>
<td><strong>Timing::Time GetTime () const</strong></td>
<td>get current time in seconds</td>
</tr>
<tr>
<td><strong>Timing::Tick GetTicks () const</strong></td>
<td>get current time in ticks</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Win360::Win360Timer::Start()

start/continue the timer

Start the timer. This will update the diffTime member to reflect the accumulated time when the timer was not running (basically the difference between this timer's time and the real system time).

void Win360::Win360Timer::Stop()

stop the timer

Stop the timer. This will record the current realtime, so that the next Start() can measure the time lost between Stop() and Start() which must be taken into account to keep track of the difference between this timer's time and realtime.

void Win360::Win360Timer::Reset()

reset the timer

Reset the timer so that will start counting at zero again.

bool Win360::Win360Timer::Running() const

return true if currently running

Returns true if the timer is currently running.

Timing::Time Win360::Win360Timer::GetTime() const

get current time in seconds
This returns the timer's current time in seconds.

```cpp
Timing::Tick
Win360::Win360Timer::GetTicks() const
```

get current time in ticks

This returns the timer's current time in "ticks". A tick is defined as one millisecond (1/1000 seconds).
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

XInput::XInputGamePad
XInput::XInputGamePad Class Reference

#include <xinputgamepad.h>

Inheritance diagram for XInput::XInputGamePad:
Detailed Description

Common gamepad support for Xbox360 and Windows.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Button</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>gamepad buttons</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th>Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>gamepad axis</td>
</tr>
</tbody>
</table>
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>XInputGamePad()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~XInputGamePad()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>bool IsConnected()</code></td>
<td>return true if this game pad is currently connected</td>
</tr>
<tr>
<td><code>void SetPlayerIndex(IndexT i)</code></td>
<td>set player index -&gt; TODO make threadsafe</td>
</tr>
<tr>
<td><code>IndexT GetPlayerIndex()</code></td>
<td>get the player index of this game pad</td>
</tr>
<tr>
<td><code>bool ButtonPressed(Button btn)</code></td>
<td>return true if a button is currently pressed</td>
</tr>
<tr>
<td><code>bool ButtonDown(Button btn)</code></td>
<td>return true if button was down at least once in current frame</td>
</tr>
<tr>
<td><code>bool ButtonUp(Button btn)</code></td>
<td>return true if button was up at least once in current frame</td>
</tr>
<tr>
<td><code>float GetAxisValue(Axis axis)</code></td>
<td>get current axis value</td>
</tr>
<tr>
<td><code>void SetLowFrequencyVibrator(float f)</code></td>
<td>set low-frequency vibration effect (0.0f .. 1.0f)</td>
</tr>
<tr>
<td><code>float GetLowFrequencyVibrator()</code></td>
<td>get low-frequency vibration</td>
</tr>
<tr>
<td><code>void SetHighFrequencyVibrator(float f)</code></td>
<td>set high-frequency vibration effect (0.0f .. 1.0f)</td>
</tr>
<tr>
<td><code>float GetHighFrequencyVibrator()</code></td>
<td>get high-frequency vibration</td>
</tr>
<tr>
<td><code>Util::Array&lt;Input::InputEvent&gt;</code> GetStateAsInputEvents()`</td>
<td>get current state as an array of input events (override in subclass!)</td>
</tr>
<tr>
<td><code>bool IsAttached()</code></td>
<td>return true if the input handler is currently attached</td>
</tr>
<tr>
<td><code>virtual void BeginCapture()</code></td>
<td></td>
</tr>
</tbody>
</table>
virtual void EndCapture ()
// end input capturing to this event handler

bool IsCapturing () const
// return true if this input handler captures input

int GetRefCount () const
// get the current refcount

void AddRef ()
// increment refcount by one

void Release ()
// decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const
// return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const
// return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
// return true if this object is instance of given class by fourcc

bool IsA (const Rtti &rtti) const
// return true if this object is instance of given class, or a derived class

bool IsA (const Util::String &rttiName) const
// return true if this object is instance of given class, or a derived class, by string

bool IsA (const Util::FourCC &rttiFourCC) const
// return true if this object is instance of given class, or a derived class, by fourcc

const Util::String & GetClassName () const
// get the class name

Util::FourCC GetClassFourCC () const
// get the class FourCC code
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Util::String</td>
<td><code>ButtonAsString (Button btn)</code></td>
<td>convert button code to string</td>
</tr>
<tr>
<td>static Util::String</td>
<td><code>AxisAsString (Axis a)</code></td>
<td>convert axis to string</td>
</tr>
<tr>
<td>static SizeT</td>
<td><code>GetMaxNumPlayers ()</code></td>
<td>get maximum number of players</td>
</tr>
<tr>
<td>static void</td>
<td><code>DumpRefCountingLeaks ()</code></td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void OnAttach()</td>
<td>called when the handler is attached to the input server</td>
</tr>
<tr>
<td>virtual void OnBeginFrame()</td>
<td>called on InputServer::BeginFrame()</td>
</tr>
<tr>
<td>void UpdateButtonState(const XINPUT_GAMEPAD &amp;curState, WORD xiBtn, Button btn)</td>
<td>update the state of a game pad button</td>
</tr>
<tr>
<td>void UpdateTriggerAxis(const XINPUT_GAMEPAD &amp;curState, Axis axis)</td>
<td>update the state of a trigger axis</td>
</tr>
<tr>
<td>void UpdateThumbAxis(const XINPUT_GAMEPAD &amp;curState, Axis axis)</td>
<td>update the state of a thumb stick axis</td>
</tr>
<tr>
<td>virtual void OnReset()</td>
<td>reset the input handler</td>
</tr>
<tr>
<td>virtual void OnRemove()</td>
<td>called when the handler is removed from the input server</td>
</tr>
<tr>
<td>virtual void OnEndFrame()</td>
<td>called on InputServer::EndFrame();</td>
</tr>
<tr>
<td>virtual void OnObtainCapture()</td>
<td>called when input handler obtains capture</td>
</tr>
<tr>
<td>virtual void OnReleaseCapture()</td>
<td>called when input handler looses capture</td>
</tr>
<tr>
<td>virtual bool OnEvent(const InputEvent &amp;inputEvent)</td>
<td>called when an input event should be processed</td>
</tr>
</tbody>
</table>
Member Function Documentation

void XInput::XInputGamePad::OnBeginFrame() [protected, virtual]
called on InputServer::BeginFrame()

This compares the current state of the game pad against the previous state and sets the internal state accordingly.

FIXME: Calling XInputGetState() on non-connected controllers is very expensive, thus if XInputGetState return ERROR_DEVICE_NOT_CONNECTED, only call XInputGetState() every 2 seconds to check if a device has actually been connected!!!

Reimplemented from Input::InputHandler.

void XInput::XInputGamePad::UpdateButtonState(const XINPUT_GAMEPAD& curState, WORD xiBtn, Button btn) [protected]

update the state of a game pad button

Comesares the previous and current state of a game pad button and updates the parent class’ state accordingly.

Array<InputEvent> Base::GamePadBase::GetStateAsInputEvents() const [inherited]

get current state as an array of input events (override in subclass!)

This method should return the current state of the game pad as input events. It is up to a specific subclass to implement this method.

void Input::InputHandler::BeginCapture() [virtual, inherited]
capture input to this event handler

Begin capturing input to this input handler. This method must be overridden in a subclass, the derived method must call ObtainMouseCapture(), ObtainKeyboardCapture(), or both, depending on what type input events you want to capture. An input handler which captures input gets all input events of the given type exclusively.

Reimplemented in Base::KeyboardBase, and Base::MouseBase.

```cpp
void Input::InputHandler::EndCapture()
```

[virtual, inherited]

discontinued use of input capturing to this event handler

End capturing input to this input handler. Override this method in a subclass and release the captures obtained in BeginCapture().

Reimplemented in Base::KeyboardBase, and Base::MouseBase.

```cpp
int Core::RefCounted::GetRefCount() const
```

[inline, inherited]

get the current refcount

Return the current refcount of the object.

```cpp
void Core::RefCounted::AddRef()
```

[inline, inherited]

increment refcount by one

Increment the refcount of the object.

```cpp
void Core::RefCounted::Release()
```

[inline, inherited]

decrement refcount and destroy object if refcount is zero

Decrement the refcount and destroy object if refcount is zero.

```cpp
const Util::String &
```
Core::RefCounted::GetClassName ( ) const [inline, inherited]

get the class name

Get the class name of the object.

**Util::FourCC**
Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]

get the class FourCC code

Get the class FourCC of the object.

void
Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)

This method should be called as the very last before an application exits.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields
The Nebula Device 3 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- AngularPFeedbackLoop
- Animation::AnimEventInfo
- Animation::AnimJobEnqueueMode
- Animation::AnimSequencer
- Animator::AnimatorInstance
- Animator::AnimLoopType
- AnimatorNode
- AnimKey
- AnimKeyArray
- App::Application
  - App::ConsoleApplication
  - App::GameApplication
  - App::RenderApplication
    - App::ViewerApplication
- Application::StateHandler
- Base::ArchiveFileSystemBase
- Base::CalendarTimeBase
  - Win360::Win360CalendarTime
  - Timing::CalendarTime
- Base::JobFuncDescBase
  - Jobs::SerialJobFuncDesc
  - Jobs::TPJobFuncDesc
- Base::ParticleSystemInstanceBase
- Base::ResourceAllocator
- Base::ResourceLump
- Base::SkinnedMeshDrawInfoBase
  - Characters::SkinnedMeshDrawInfo
- Base::SystemInfoBase
  - Win32::Win32SystemInfo
    - System::SystemInfo
- Base::TextureBase::MapInfo
- BaseGameFeature::CategoryManager::Category
- `BaseGameFeature::CategoryManager::Entry`
- `BaseGameFeature::EntityManager::DelayedJob`
- `BaseGameFeature::FactoryManager::PropertyEntry`
- `BaseGameFeature::GameStateHandler`
- `BaseGameFeature::LevelLoader`
- `BaseGameFeature::TimeProperty`
- `BaseGameFeature::TimingTimeSource`
- `Characters::CharacterAnimationController`
- `Characters::CharacterJoint`
- `Characters::CharacterSkeleton`
- `Characters::CharacterSkeletonInstance`
- `Characters::CharacterSkin`
- `Characters::CharacterSkinLibrary`
- `Characters::CharacterSkinList`
- `Characters::CharacterSkinSet`
- `Characters::CharacterVariationSet`
- `Characters::CharJointComponents`
- `Characters::SkinningTechnique`
- `Commands::CreateEntityCommand`
- `Conditions::Operator< TYPE >`
- `core`
- `Core::ExitHandler`
- `Core::Factory`
- `Core::RefCounted`
  - `Animation::AnimJob`
    - `Animation::PlayClipJob`
  - `Base::DisplayDeviceBase`
    - `Win32::Win32DisplayDevice`
      - `Direct3D9::D3D9DisplayDevice`
  - `CoreGraphics::DisplayDevice`
  - `Base::GameContentServerBase`
    - `IO::GameContentServer`
  - `Base::InputServerBase`
    - `Win32::Win32InputServer`
      - `Input::InputServer`
  - `Base::JobBase`
    - `Jobs::SerialJob`
    - `Jobs::TPJob`
    - `Jobs::Job`
- `Base::JobPortBase`
  - `Jobs::SerialJobPort`
  - `Jobs::TPJobPort`
    - `Jobs::JobPort`
- `Base::MouseRenderDeviceBase`
  - `CoreGraphics::MouseRenderDevice`
- `Base::MultipleRenderTargetBase`
  - `CoreGraphics::MultipleRenderTarget`
- `Base::ParticleRendererBase`
  - `Direct3D9::D3D9ParticleRenderer`
    - `Particles::ParticleRenderer`
- `Base::RenderDeviceBase`
  - `Direct3D9::D3D9RenderDevice`
    - `CoreGraphics::RenderDevice`
- `Base::RenderTargetBase`
  - `Direct3D9::D3D9RenderTarget`
    - `CoreGraphics::RenderTarget`
- `Base::ShaderInstanceBase`
  - `Direct3D9::D3D9ShaderInstance`
    - `CoreGraphics::ShaderInstance`
- `Base::ShaderServerBase`
  - `Direct3D9::D3D9ShaderServer`
    - `CoreGraphics::ShaderServer`
- `Base::ShaderVariableBase`
  - `Direct3D9::D3D9ShaderVariable`
    - `CoreGraphics::ShaderVariable`
- `Base::ShaderVariableInstanceBase`
  - `CoreGraphics::ShaderVariableInstance`
- `Base::ShaderVariationBase`
  - `Direct3D9::D3D9ShaderVariation`
    - `CoreGraphics::ShaderVariation`
- `Base::ShapeRendererBase`
  - `Win360::D3D9ShapeRenderer`
    - `CoreGraphics::ShapeRenderer`
- `Base::SkinnedMeshRendererBase`
  - `Win32::Win32SkinnedMeshRenderer`
    - `Characters::SkinnedMeshRenderer`
- `Base::TextRendererBase`
  - `Direct3D9::D3D9TextRenderer`
- CoreGraphics::TextRenderer
- Base::TransformDeviceBase
  - Win360::D3D9TransformDevice
  - CoreGraphics::TransformDevice
- Base::VertexLayoutBase
  - Win360::D3D9VertexLayout
  - CoreGraphics::VertexLayout
- Base::VertexLayoutServerBase
  - CoreGraphics::VertexLayoutServer
- BaseGameFeature::EntityLoaderBase
  - BaseGameFeature::EntityLoader
  - BaseGameFeature::EnvironmentLoader
- BaseGameFeature::LoaderServer
- BaseGameFeature::TimeSource
  - BaseGameFeature::InputTimeSource
  - BaseGameFeature::SystemTimeSource
- BaseGameFeature::UserProfile
- Characters::Character
- Characters::CharacterInstance
- Characters::CharacterServer
- Core::CoreServer
- CoreAnimation::AnimKeyBuffer
- CoreAnimation::AnimSampleBuffer
- CoreGraphics::DisplayEventHandler
  - CoreGraphics::ThreadSafeDisplayEventHandler
    - Win32::Win32InputDisplayEventHandler
- CoreGraphics::RenderEventHandler
  - CoreGraphics::ThreadSafeRenderEventHandler
- Debug::DebugCounter
- Debug::DebugServer
- Debug::DebugShapeRenderer
- Debug::DebugTextRenderer
- Debug::DebugTimer
- Frame::FrameBatch
- Frame::FramePassBase
  - Frame::FramePass
  - Frame::FramePostEffect
- Frame::FrameServer
- Frame::FrameShader
- FrameSync::FrameSyncSharedData
- FrameSync::FrameSyncTimer
- FSM::State
- FSM::StateMachine
- FSM::Transition
- Game::Entity
- Game::FeatureUnit
  - ScriptFeature::ScriptFeatureUnit
- Game::GameServer
- Graphics::Display
- Graphics::GraphicsEntity
  - Graphics::AbstractLightEntity
    - Graphics::GlobalLightEntity
    - Graphics::PointLightEntity
    - Graphics::SpotLightEntity
  - Graphics::CameraEntity
  - Graphics::ModelEntity
- Graphics::GraphicsServer
- Graphics::MouseRenderer
- Graphics::View
- Http::HttpRequestHandler
  - Debug::ConsolePageHandler
  - Debug::CorePageHandler
  - Debug::DebugPageHandler
  - Debug::DisplayPageHandler
  - Debug::GraphicsPageHandler
  - Debug::HelloWorldRequestHandler
  - Debug::IoPageHandler
  - Debug::MemoryPageHandler
  - Debug::MeshPageHandler
  - Debug::ObjectInspectorHandler
  - Debug::ShaderPageHandler
  - Debug::StreamingTexturePageHandler
  - Debug::SvgTestPageHandler
  - Debug::TexturePageHandler
  - Debug::ThreadPageHandler
  - Http::DefaultHttpRequestHandler
- Http::HttpServer
- Http::HttpServerProxy
- Input::InputHandler
  - Base::GamePadBase
    - XInput::XInputGamePad
    - Input::GamePad
  - Base::KeyboardBase
    - Input::Keyboard
  - Base::MouseBase
    - Win32::Win32Mouse
    - Input::Mouse
- InternalGraphics::AttachmentServer
- InternalGraphics::InternalGraphicsEntity
  - InternalGraphics::InternalCameraEntity
  - InternalGraphics::InternalModelEntity
  - Lighting::InternalAbstractLightEntity
    - Lighting::InternalGlobalLightEntity
    - Lighting::InternalPointLightEntity
    - Lighting::InternalSpotLightEntity
- InternalGraphics::InternalGraphicsServer
- InternalGraphics::InternalStage
- InternalGraphics::InternalView
- IO::ArchiveBase
  - IO::ZipArchive
    - IO::Archive
- IO::AssignRegistry
- IO::Console
- IO::ConsoleHandler
  - IO::HistoryConsoleHandler
  - IO::LogFileConsoleHandler
  - OSX::OSXConsoleHandler
  - Win32::Win32ConsoleHandler
- IO::IoServer
- IO::SchemeRegistry
- IO::Stream
  - IO::FileStream
  - IO::MemoryStream
  - IO::ZipFileStream
- IO::StreamReader
  - Http::HttpRequestReader
  - Http::HttpResponseReader
- IO::BinaryReader
- IO::BinaryWriter
- IO::BinaryXMLReader
- IO::BinaryWriter
- IO::TextReader
- IO::TextWriter
- IO::XmlReader
- IO::XmlWriter
- Messaging::MessageReader
- Messaging::MessageWriter
- Lighting::LightServerBase
- Lighting::LightPrePassServer
- Lighting::LightServer
- Lighting::SM30LightServer
- Messaging::AsyncPort
- Interface::InterfaceBase
  - Debug::DebugInterface
  - Graphics::GraphicsInterface
  - Http::HttpInterface
- Messaging::Handler
  - Animation::AnimEventHandlerBase
  - Animation::AnimEventServer
  - Interface::InterfaceHandlerBase
    - Debug::DebugGraphicsHandler
    - Debug::DebugHandler
    - Graphics::GraphicsHandler
    - Http::HttpMessageHandler
    - IO::IoInterfaceHandler
- Messaging::Message
- Actions::Action
  - Actions::ActionList
  - Actions::FSMAction
  - Actions::IfThenElseAction
- Actions::SequenceAction
- Conditions::Condition
  - Conditions::And
  - Conditions::FSMCondition
  - Conditions::Not
  - Conditions::Or
- Http::HttpRequest
- Messaging::BatchMessage
- Messaging::Port
- Game::Property
  - BaseGameFeature::TransformableProperty
    - PhysicsFeature::PhysicsProperty
      - PhysicsFeature::ActorPhysicsProperty
    - PhysicsFeature::TriggerProperty
  - GraphicsFeature::CameraProperty
    - GraphicsFeature::ChaseCameraProperty
    - GraphicsFeature::MayaCameraProperty
  - GraphicsFeature::GraphicsProperty
    - GraphicsFeature::ActorGraphicsProperty
    - GraphicsFeature::AnimationControlProperty
    - StateObjectFeature::StateGraphicsProperty
  - PhysicsFeature::EnvironmentCollideProperty
  - StateObjectFeature::StateProperty
- Messaging::Dispatcher
- Game::Manager
  - BaseGameFeature::AnimEventManager
  - BaseGameFeature::CategoryManager
  - BaseGameFeature::EntityManager
  - BaseGameFeature::EnvEntityManager
  - BaseGameFeature::EnvQueryManager
  - BaseGameFeature::FactoryManager
  - BaseGameFeature::FocusManager
  - BaseGameFeature::GlobalAttrsManager
  - BaseGameFeature::TimeManager
  - GraphicsFeature::AttachmentManager
  - Script::DialogManager
  - Script::ScriptManager
  - Script::SubstitutionManager
- Models::ModelInstance
- **Base::ResourceBase**
  - **Base::IndexBufferBase**
    - **Win360::D3D9IndexBuffer**
    - **CoreGraphics::IndexBuffer**
  - **Base::TextureBase**
    - **Direct3D9::D3D9Texture**
    - **CoreGraphics::Texture**
  - **Base::VertexBufferBase**
    - **Win360::D3D9VertexBuffer**
    - **CoreGraphics::VertexBuffer**
  - **Base::ShaderBase**
    - **Direct3D9::D3D9Shader**
    - **CoreGraphics::Shader**
  - **CoreAnimation::AnimResource**
  - **Models::Model**
- **Resources::ResourceDictionary**
- **Resources::ResourceLoader**
  - **Base::MemoryIndexBufferLoaderBase**
    - **Win360::D3D9MemoryIndexBufferLoader**
    - **CoreGraphics::MemoryIndexBufferLoader**
  - **Base::MemoryVertexBufferLoaderBase**
    - **Win360::D3D9MemoryVertexBufferLoader**
    - **CoreGraphics::MemoryVertexBufferLoader**
  - **CoreGraphics::MemoryMeshLoader**
  - **Models::StreamModelLoader**
  - **Resources::StreamResourceLoader**
    - **CoreAnimation::StreamAnimationLoader**
    - **Direct3D9::D3D9StreamShaderLoader**
    - **CoreGraphics::StreamShaderLoader**
    - **Direct3D9::D3D9StreamTextureLoader**
    - **CoreGraphics::StreamTextureLoader**
    - **Resources::D3D9TextureStreamer**
    - **Resources::TextureStreamer**
    - **Win360::D3D9StreamMeshLoader**
    - **CoreGraphics::StreamMeshLoader**
  - **Resources::ResourceManager**
  - **Resources::ResourceMapper**
    - **Resources::SimpleResourceMapper**
  - **Resources::ResourceSaver**
- Base::StreamTextureSaverBase
  - Win360::D3D9StreamTextureSaver
    - CoreGraphics::StreamTextureSaver
- Resources::ResourceScheduler
  - Resources::TexturePoolMapperScheduler
- Script::ActionReader
- Script::Dialog
- Script::DialogDesc
- Script::DialogTake
- Script::InfoLog
- Script::InfoLogFilter
- Script::Task
- Threading::ObjectRef
- Visibility::ObserverContext
- Visibility::VisibilityContainer
- Visibility::VisibilityContext
- Visibility::VisibilityQuery
- Visibility::VisibilitySystemBase
  - Visibility::VisibilityBoxSystem
  - Visibility::VisibilityClusterSystem
  - Visibility::VisibilityQuadtree
- Win360::Win360Socket
  - Net::Socket
- Win360::Win360Thread
  - Threading::Thread
    - Jobs::TPWorkerThread
    - Messaging::HandlerThreadBase
      - FrameSync::FrameSyncHandlerThread
      - Messaging::BlockingHandlerThread
      - Messaging::RunThroughHandlerThread
- Core::Rtti
- Core::Singleton
- Core::SysFunc
- CoreAnimation::AnimClip
- CoreAnimation::AnimCurve
- CoreAnimation::AnimEvent
- CoreAnimation::AnimEventEmitter
- CoreAnimation::AnimSampleMixInfo
- CoreAnimation::AnimUtil
- Http::Http NzStream
- Http::HttpStatus
- Http::HttpStream
- Input::InputEvent
- Input::InputPriority
- Input::Key
- Input::MouseButton
- InternalGraphics::InternalGraphicsEntityType
- InternalGraphics::VisibilityCell
- IO::BXmlLoaderUtil
- IO::FileTime
- IO::Interface
- IO::MediaType
- IO::URI
- IO::ZipDirEntry
- IO::ZipFileEntry
- IO::ZipFileSystem
  - IO::ArchiveFileSystem
- Jobs::JobContext
- Jobs::JobDataDesc
- Jobs::JobFuncDesc
- Jobs::JobServerBase
- Jobs::JobUniformDesc
- Jobs::SerialJobSystem
- Jobs::TPJobCommand
- Jobs::TPJobSlice
- Jobs::TPJobSystem
  - Jobs::JobSystem
- Jobs::TPJobThreadPool
- Legacy::Nvx2StreamReader
- Lighting::LightType
- Lighting::PSSMUtil
- Lighting::SM30ShadowServer
  - Lighting::ShadowServer
- Managers::LogManager
- Managers::QuestManager
- Managers::ScriptTable
- Managers::ScriptTemplateManager
- Math::bbox
- Math::ClipStatus
- Math::Extrapolator< TYPE >
- Math::float2
- Math::float4
  - Math::point
  - Math::vector
- Math::frustum
- Math::line
- Math::matrix44
- Math::noise
- Math::plane
- Math::polar
- Math::quaternion
- Math::rectangle< TYPE >
- Math::sphere
- Math::transform44
- Memory::Heap
- Memory::Memory
- Memory::MemoryPool
- Memory::PoolArrayAllocator
- Memory::TotalMemoryStatus
- Messaging::DelegateTable
- Messaging::Id
- Messaging::StaticMessageHandler
- MGraphics::VisibilityBox
- MGraphics::VisibilityCluster
- Models::ModelNodeType
- Models::VisResolveContainer< TYPE >
- Net::DebugMessage
- Net::DebugPacket
- Net::IpAddress
- Net::TcpMessageCodec
- OSX::OSXCpu
- OSX::OSXCriticalSection
- OSX::OSXFileTime
- OSX::OSXFSWrapper
- OSX::OSXGuid
- OSX::OSXHeap
- OSX::OSXInterlocked
- OSX::OSXMemoryPool
- OSX::OSXThreadId
- OSX::OSXThreadLocalPtr< TYPE >
- OSX::SysFunc
- Particles::EmitterAttrs
- Particles::EmitterMesh
- Particles::EnvelopeCurve
- Particles::EnvelopeSampleBuffer
- Particles::Particle
- Particles::ParticleRenderInfo
- Particles::ParticleSystemRenderer
- Particles::ParticleSystemState
- PFeedbackLoop
- PhysicsFeature::MouseGripperProperty
- PIDFeedbackLoop
- PriorityArray
- Properties::AmbienceBubbleProperty
- Properties::BehaviourProperty
- Properties::CutsceneCameraProperty
- Properties::EnvironmentGraphicsProperty
- Properties::InputProperty
- Properties::LightProperty
- Properties::PathAnimProperty
- Ptr< TYPE >
- QuadTree
- Render::RenderConsoleHandler
- RenderUtil::DrawFullScreenQuad
- RenderUtil::MayaCameraUtil
- RenderUtil::MouseRayUtil
- RenderUtil::NodeLookupUtil
- Resources::PoolResourceMapper
- Resources::PoolSetupInfo
- Resources::ResourceDictionary::Entry
- Resources::ResourceId
- Round
- Script::Attributes
- Shared::CameraSettings
- Shared::CharJointInfo
- Shared::GraphicsEntityShared
- Shared::ModelEntityShared
- StateObjectFeature::StateInfo
- StateObjectFeature::StateObjectFeatureUnit
- System::AppEntry
- System::ByteOrder
- System::Cpu
- System::Win32Environment
- System::Win32Registry
- Threading::Barrier
- Threading::SafeFlag
- Util::Array< TYPE >
- Util::Array< Util::KeyValuePair< PRITYPE, TYPE > >
- Util::BitField< NUMBITS >
- Util::Blob
- Util::CommandLineArgs
- Util::Crc
- Util::Delegate< ARGTYPE >
- Util::Dictionary< KEYTYPE, VALUETYPE >
- Util::FixedArray< TYPE >
- Util::FixedTable< TYPE >
- Util::FourCC
- Util::GlobalStringAtomTable::DebugInfo
- Util::Guid
- Util::HashTable< KEYTYPE, VALUETYPE >
- Util::KeyValuePair< KEYTYPE, VALUETYPE >
- Util::KeyValuePair< PRITYPE, TYPE >
- Util::KeyValuePair< Util::String, Util::String >
  - IO::Assign
- Util::List< TYPE >
  - Core::RefCountedList
- Util::List< TYPE >::Iterator
- Util::List< Core::RefCounted * >
- Util::QuadTree< TYPE >::Node
- Util::Queue< TYPE >
  - Threading::SafeQueue< TYPE >
- Util::Queue< Util::KeyValuePair< PRITYPE, TYPE > >
  - Threading::SafePriorityQueue< PRITYPE, TYPE >
- Util::RandomNumberTable
- Util::RingBuffer< TYPE >
- Util::RunLengthCodec
- Util::SimpleTree< VALUETYPE >
- Util::SimpleTree< VALUETYPE >::Node
- Util::SparseTable< TYPE >
- Util::Stack< TYPE >
- Util::String
- Util::StringAtom
- Util::StringAtomTableBase
  - Util::GlobalStringAtomTable
  - Util::LocalStringAtomTable
- Util::StringAtomTableBase::StaticString
- Util::StringBuffer
- Util::Variant
- Visibility::VisibilityChecker
- WeakPtr< TYPE >
- Win32::SysFunc
- Win32::Win32Cpu
- Win32::Win32Guid
- Win32::Win32MiniDump
  - Debug::MiniDump
- Win32::Win32StringConverter
- Win360::D3D9Types
- Win360::Win360Barrier
- Win360::Win360CriticalSection
  - Threading::CriticalSection
- Win360::Win360Event
  - Threading::Event
- Win360::Win360FileTime
- Win360::Win360FSWrapper
  - IO::FSWrapper
- Win360::Win360Heap
- Win360::Win360Interlocked
  - Threading::Interlocked
- Win360::Win360IpAddress
- Win360::Win360MemoryPool
- Win360::Win360ThreadBarrier
  - Threading::ThreadBarrier
- Win360::Win360Threadld
- Win360::Win360Timer
■ **Timing::Timer**

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:40 2010
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- __declspec() : Math::float4, Math::plane
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
## The Nebula Device 3 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<table>
<thead>
<tr>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
</tr>
<tr>
<td>Application</td>
</tr>
<tr>
<td>AsyncGraphics</td>
</tr>
<tr>
<td>Base</td>
</tr>
<tr>
<td>BaseGameFeature</td>
</tr>
<tr>
<td>Characters</td>
</tr>
<tr>
<td>Core</td>
</tr>
<tr>
<td>CoreAnimation</td>
</tr>
<tr>
<td>CoreGraphics</td>
</tr>
<tr>
<td>Debug</td>
</tr>
<tr>
<td>Direct3D9</td>
</tr>
<tr>
<td>Frame</td>
</tr>
<tr>
<td>Game</td>
</tr>
<tr>
<td>Graphics</td>
</tr>
<tr>
<td>GraphicsFeature</td>
</tr>
<tr>
<td>Http</td>
</tr>
<tr>
<td>Input</td>
</tr>
<tr>
<td>Interface</td>
</tr>
<tr>
<td>IO</td>
</tr>
<tr>
<td>Legacy</td>
</tr>
<tr>
<td>Lighting</td>
</tr>
<tr>
<td>Loader</td>
</tr>
<tr>
<td>Math</td>
</tr>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Messaging</td>
</tr>
<tr>
<td>Models</td>
</tr>
<tr>
<td>Net</td>
</tr>
<tr>
<td>PhysicsFeature</td>
</tr>
<tr>
<td>RenderUtil</td>
</tr>
<tr>
<td>Resources</td>
</tr>
<tr>
<td>ScriptFeature</td>
</tr>
<tr>
<td>StateObjectFeature</td>
</tr>
<tr>
<td>System</td>
</tr>
<tr>
<td>Threading</td>
</tr>
<tr>
<td>Timing</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>Util</td>
</tr>
<tr>
<td>Visibility</td>
</tr>
<tr>
<td>Win32</td>
</tr>
<tr>
<td>Win360</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by `doxygen` at Fri Mar 26 15:21:40 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
App Namespace Reference
Detailed Description

The Nebula3 App Subsystem

The **App** namespace offers a set of **Application** classes which simplify setting up a proper Nebula3 application. The general idea is that an application derives a subclass from one of the specialized **Application** classes and adds its own functionality to the virtual **Application::Open()**, **Application::Run()** and **Application::Close()** methods.

Here's an example of what a Nebula3's main source code should look like:

```cpp
#include "stdneb.h"
#include "apprender/viewerapplication.h"

using namespace App;
using namespace Util;

ImplementNebulaApplication();

/**
 * void
 * NebulaMain(const CmdLineArgs& args)
 * {
 *     ViewerApplication app;
 *     app.SetCompanyName("Radon Labs GmbH");
 *     app.SetAppName("nViewer3");
 *     app.SetCmdLineArgs(args);
 *     if (app.Open())
 *     {
 *         app.Run();
 *         app.Close();
 *     }
 *     app.Exit();
 * }
```
The macro `ImplementNebulaApplication()` takes care about some platform-specifics (mainly how arguments are passed to a program) and then calls the `NebulaMain()` function, which receives a `Util::CmdLineArgs` object which contains the command line arguments. The `ViewerApplication` class is a user-derived class (in this case Nebula3's standard viewer). The application object needs to be setup with a company name, an application name (these two uniquely identify the application and are for used to create a data directory under "My Files" which will contain application specific files (likes configuration settings or save game files). The `Open()` method will setup the application for use. If something goes wrong the method will return false. `Run()` should implement the actual application features, it may run in a loop until the user wants to exit, or it may return immediately. `Close()` will shutdown the application. Finally `Exit()` must be called to properly cleanup Nebula3 before exiting the application. This will shutdown any static objects, perform a RefCounting leak and memory leak check and finally exit the application process.
### Data Structures

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>ConsoleApplication</td>
<td></td>
</tr>
<tr>
<td>RenderApplication</td>
<td></td>
</tr>
<tr>
<td>ViewerApplication</td>
<td></td>
</tr>
<tr>
<td>GameApplication</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:40 2010
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
Application Namespace Reference
Detailed Description

The **Application** namespace contains all base classes for game logic.
Data Structures

class StateHandler
Functions

void Exit ()
**Function Documentation**

```cpp
void Application::Exit()
```

This method must be called right before the main() function's end. It will properly cleanup the Nebula3 runtime, its static objects, private heaps and finally produce a refcount leak and mem leak report (debug builds only).
<table>
<thead>
<tr>
<th>home</th>
<th>namespace list</th>
</tr>
</thead>
</table>

- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
AsyncGraphics Namespace Reference
Detailed Description

FIXME!
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
Base Namespace Reference
Detailed Description

The Nebula3 Base Namespace

The **Base** subsystem contains all base classes where platformspecific subclasses are derived from. Nebula3 uses a compile-time-approach for portable code whenever possible. Platform-specific classes are often derived from a common base class, and then derived into a platform-neutral class name. For instance, the `CoreGraphics::RenderDevice` class may be (conditionally) derived from `Xbox360::Xbox360RenderDevice` or from `Direct3D::D3D9RenderDevice`, which in turn are derived from the class `Base::RenderDeviceBase` which contains common functionalty of all RenderDevice classes.

The main reason why the **Base** namespace has been introduced is to not contanimate the autodocs and IntelliSense with unrelated information.
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GameContentServerBase</code></td>
</tr>
<tr>
<td><code>JobBase</code></td>
</tr>
<tr>
<td><code>JobFuncDescBase</code></td>
</tr>
<tr>
<td><code>JobPortBase</code></td>
</tr>
<tr>
<td><code>SystemInfoBase</code></td>
</tr>
<tr>
<td><code>CalendarTimeBase</code></td>
</tr>
<tr>
<td><code>SkinnedMeshDrawInfoBase</code></td>
</tr>
<tr>
<td><code>SkinnedMeshRendererBase</code></td>
</tr>
<tr>
<td><code>DisplayDeviceBase</code></td>
</tr>
<tr>
<td><code>IndexBufferBase</code></td>
</tr>
<tr>
<td><code>MemoryIndexBufferLoaderBase</code></td>
</tr>
<tr>
<td><code>MemoryVertexBufferLoaderBase</code></td>
</tr>
<tr>
<td><code>MeshBase</code></td>
</tr>
<tr>
<td><code>MouseRenderDeviceBase</code></td>
</tr>
<tr>
<td><code>MultipleRenderTargetBase</code></td>
</tr>
<tr>
<td><code>RenderTargetBase</code></td>
</tr>
<tr>
<td><code>ResourceBase</code></td>
</tr>
<tr>
<td><code>ShaderBase</code></td>
</tr>
<tr>
<td><code>ShaderInstanceBase</code></td>
</tr>
<tr>
<td><code>ShaderServerBase</code></td>
</tr>
<tr>
<td><code>ShaderVariableBase</code></td>
</tr>
<tr>
<td><code>ShaderVariableInstanceBase</code></td>
</tr>
<tr>
<td><code>ShaderVariationBase</code></td>
</tr>
<tr>
<td><code>ShapeRendererBase</code></td>
</tr>
<tr>
<td><code>StreamTextureSaverBase</code></td>
</tr>
<tr>
<td><code>TextRendererBase</code></td>
</tr>
<tr>
<td><code>TextureBase</code></td>
</tr>
<tr>
<td><code>TransformDeviceBase</code></td>
</tr>
</tbody>
</table>
class VertexBufferBase
class VertexLayoutBase
class VertexLayoutServerBase
class GamePadBase
class InputServerBase
class KeyboardBase
class MouseBase
class ParticleRendererBase
class ArchiveFileSystemBase
class ParticleSystemInstanceBase
class ResourceAllocator
class ResourceLump
BaseGameFeature Namespace
Reference
Detailed Description

The **BaseGameFeature** offers classes for loading and saving entities from and to the database. It allows creating of entities from categories defined in the blueprint.xml (entity and factorymanager). Furthermore the basegamefeature creates managers for focusing entities, using global attributes (from globals.xml), creating enviornment entities.
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputTimeSource</td>
</tr>
<tr>
<td>SystemTimeSource</td>
</tr>
<tr>
<td>TimeManager</td>
</tr>
<tr>
<td>TimeSource</td>
</tr>
<tr>
<td>EntityLoader</td>
</tr>
<tr>
<td>EntityLoaderBase</td>
</tr>
<tr>
<td>EnvironmentLoader</td>
</tr>
<tr>
<td>LevelLoader</td>
</tr>
<tr>
<td>LoaderServer</td>
</tr>
<tr>
<td>UserProfile</td>
</tr>
<tr>
<td>AnimEventManager</td>
</tr>
<tr>
<td>CategoryManager</td>
</tr>
<tr>
<td>EntityManager</td>
</tr>
<tr>
<td>EnvEntityManager</td>
</tr>
<tr>
<td>EnvQueryManager</td>
</tr>
<tr>
<td>FactoryManager</td>
</tr>
<tr>
<td>FocusManager</td>
</tr>
<tr>
<td>GlobalAttrsManager</td>
</tr>
<tr>
<td>TransformableProperty</td>
</tr>
<tr>
<td>GameStateHandler</td>
</tr>
<tr>
<td>TimingTimeSource</td>
</tr>
<tr>
<td>TimeProperty</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
Characters Namespace Reference
Detailed Description

FIXME!
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
</tr>
<tr>
<td>CharacterAnimationController</td>
</tr>
<tr>
<td>CharacterInstance</td>
</tr>
<tr>
<td>CharacterJoint</td>
</tr>
<tr>
<td>CharacterNode</td>
</tr>
<tr>
<td>CharacterNodeInstance</td>
</tr>
<tr>
<td>CharacterServer</td>
</tr>
<tr>
<td>CharacterSkeleton</td>
</tr>
<tr>
<td>CharacterSkeletonInstance</td>
</tr>
<tr>
<td>CharacterSkin</td>
</tr>
<tr>
<td>CharacterSkinLibrary</td>
</tr>
<tr>
<td>CharacterSkinList</td>
</tr>
<tr>
<td>CharacterSkinNode</td>
</tr>
<tr>
<td>CharacterSkinNodeInstance</td>
</tr>
<tr>
<td>CharacterSkinSet</td>
</tr>
<tr>
<td>CharacterVariationSet</td>
</tr>
<tr>
<td>CharJointComponents</td>
</tr>
<tr>
<td>SkinnedMeshDrawInfo</td>
</tr>
<tr>
<td>SkinnedMeshRenderer</td>
</tr>
<tr>
<td>SkinningTechnique</td>
</tr>
</tbody>
</table>
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
Core Namespace Reference
Detailed Description

The Nebula3 Core Subsystem

The Nebula3 Core subsystem (as the name implies) implements the core concepts of Nebula3 which are:

- a RefCounted base class which implements a strong ref counting mechanism
- a runtime type information system
- a templated smart pointer class Ptr<> which manages the life time of RefCounted objects
- a factory mechanism which allows to construct C++ objects from their string class name
- a central Server object which sets up a basic Nebula3 runtime environment

The Nebula3 Object Model

Nebula3 implements a basic object model which implements the following new features on top of the C++ object model:

- lifetime management by refcounting and smart pointers
- object creation by string or fourcc class identifier
- a runtime type information system

Implementing A New Nebula3 Class

The first decision when implementing a new class should be whether the new class should be derived from the Core::RefCounted class or whether it should be a traditional C++ class. The following points should help to find the answer:

- if the class wants to make use of the extended Nebula3 object model features like refcounting, RTTI, and so forth, it must be derived from the Core::RefCounted class
- if the class is a typical small helper or utility class, like a dynamic
array class, a math vector class, or something similar, it often does not make sense to derive from `Core::RefCounted`.

Deriving from the `Core::RefCounted` class implies some restrictions:

- RefCounted-derived objects may never be created directly in the local C++ context as stack objects, since stack objects are lifetime-managed by C++ (they are destroyed when the current C++ context is left, circumventing Nebula3’s refcounted lifetime management completely).
- RefCounted-derived classes only have a default constructor.
- RefCounted-derived classes must have a virtual destructor.
- RefCounted-derived classes must not be copied, since this would confuse the refcounting mechanism.

To make use of the Nebula3 object model features, one needs to derive from the `Core::RefCounted` class and annotate the new class with some additional information in the class declaration and in the header file:

A normal RefCounted-derived class declaration usually looks like this:

```cpp
namespace MyNamespace
{
    class MyClass : public Core::RefCounted
    {
        DeclareClass(MyClass);
        public:
            /// constructor
            MyClass();
            /// destructor
            virtual ~MyClass();
    }
    RegisterClass(MyClass);
}
```

Notice the DeclareClass() macro, the default constructor and the virtual destructor and the RegisterClass() macro outside of the class declaration. The DeclareClass() macro adds some minimal Nebula3-specific information to the class declaration for the RTTI and factory mechanism. The DeclareClass() macro generally hides the internals of
the Nebula3 object model from the programmer, so that (hopefully), internals of the object model can be changed without affecting existing classes. The RegisterClass() macro is optional and registers the class with the central factory object. If you know that objects of this class will never be created by string class name or fourcc code, the RegisterClass() macro can be omitted.

The .cc side of the class needs to contain the following Nebula3 specific information:

```cpp
namespace MyNamespace
{
    ImplementClass(MyNamespace::MyClass, 'MYCL', Core::RefCounted);
}
```

The ImplementClass() macro registers the class with the RTTI mechanism, the first parameter describes the C++ class name (note that the namespace must be present here. The second macro is the class fourcc code, which must be unique across all classes (you'll get a runtime error at application startup if 2 classes try to register with the same fourcc code). The third argument is the C++ class name of the parent class. This is used by the RTTI mechanism to reconstruct the class tree.

**RefCounting And Smart Pointers**

Nebula3 uses traditional refcounting to manage the lifetime of its objects. A templated smart pointer class Ptr<> exists to hide the refcounting details from the programmer. As a general rule of thumb, always use smart pointers to point to RefCounted-derived objects unless you can make sure that within a given code block, the refcount of an object will not change.

Smart pointers have a number of advantages over plain pointers:

- accessing a 0-pointer will give you an easy to debug assertion instead of a memory fault
- you'll never have to call AddRef() or Release() on you refcounted
objects (in fact if you have, there's something seriously wrong)

- smart pointers work nicely in container classes, an array of smart pointers instead of plain pointers eliminates all the typical lifetime management problems, you never need to take care about releasing the objects behind the pointers, instead the array just behaves like it would contain real C++ objects
- with smart pointers, you generally don't need to define "object ownership" as is often the case when using plain pointers (who's responsible to delete objects, and so on...)

There are also some disadvantages with smart pointers:

- Performance: Copying and assigning smart pointers involves calling AddRef() and/or Release() on their objects, de-referencing a smart pointer involves an assertion-check that the contained object pointer is valid. The resulting performance hit is usually negligible, but you may have to be aware of it in inner loops.
- Presumably dead objects still alive: Since objects managed by smart pointers are only deleted when the last client gives up ownership, objects may exist longer then intended. Often this is points to a bug. Nebula3 will notify you about any refcounting leaks (that is, refcounting objects that still exist at application shutdown)

Creating Nebula3 Objects

Nebula3 objects that are derived from Core::RefCounted can be created in 3 different ways:

Directly through the static create method:

```cpp
Ptr<MyClass> myObj = MyClass::Create();
```

The static Create() method is added to the class through the DeclareClass() macro described before. This is basically just syntactic sugar for the C++ operator::new(). In fact, the Create() method is nothing more then an inline method with a call to the new operator inside. Also note the correct use of a smart pointer to hold the new object.
Another way to create a Nebula3 method is by class name:

```cpp
using namespace Core;
Ptr<MyClass> myObj = (MyClass*) Factory::Instance()->Create("My
```

Creating an object by its string class name is useful if you don't know the object class at compile time, which is usually the case when serialized objects are restored, or when some sort of scripting interface is used. Note the type cast. This is necessary because the factory Create() method returns a generic pointer to a 

```cpp
Core::RefCounted
```

object.

A variation of the create-by-class-name method is to create the object by its class fourcc code:

```cpp
using namespace Core;
using namespace Util;
Ptr<MyClass> myObj = (MyClass*) Factory::Instance()->Create(Fou
```

This method looks less intuitive, but it is often faster as create-by-name and the fourcc class identifier uses less space (4 bytes) then the string class name, which may be of advantage when objects are encoded/decoded to and from binary streams.

**The Nebula3 Runtime Type Information System**

The Nebula3 RTTI system gives you access to an objects class type at runtime and lets you check whether an object is the exact instance of a class, or an instance of a derived class. You can also get the class name or the class fourcc identifier directly from an object. All this functionality is implemented behind the scenes in the DeclareClass() and ImplementClass() macros. The RTTI mechanism is more efficient and easier to use then the RTTI mechanism in Nebula1 and Nebula2.

Here's some example code:

```cpp
using namespace Util;
using namespace Core;
```
// check whether an object is instance of a specific class
if (myObj->IsInstanceOf(MyClass::RTTI))
{
    // it's a MyClass object
}

// check whether an object is instance of a derived class
if (myObj->IsA(RefCounted::RTTI))
{
    // it's a RefCounted instance or some RefCounted-derived in
}

// get the class name of my object, this yields "MyNamespace::MyClass"
const String& className = myObj->GetClassName();

// get the fourcc class identifier of my object, this yields 'M
const FourCC& fourcc = myObj->GetClassFourCC();

You can also query the central factory object whether a given class
has been registered:

using namespace Core;

// check if a class has been registered by class name
if (Factory::Instance()->ClassExists("MyNamespace::MyClass"))
{
    // yep, the class exists
}

// check if a class has been registered by class fourcc code
if (Factory::Instance()->ClassExists(FourCC('MYCL')))  
{
    // yep, the class exists
}

**Nebula3 Singletons**

Many central Nebula3 objects are singletons, that is, an object which
only exists once in the application and often is known to all other
objects in the application.

Access to singleton objects can be gained through the static
Instance() method, which returns a pointer to the single instance of the
singleton class. The returned pointer is guaranteed to be valid. If the singleton object doesn't exist at the time the Instance() method is called, an assertion will be thrown:

```cpp
// obtain a pointer to the Core::Server singleton
Ptr<Core::Server> coreServer = Core::Server::Instance();
```

You can also check for the existance of a given singleton:

```cpp
// does the Core::Server object exist?
if (Core::Server::HasInstance())
{
    // yep, the core server exists
}
```

Nebula3 provides some helper macros to implement a singleton class:

```cpp
// declare a singleton class
class MySingletonClass : public Core::RefCounted
{
    DeclareClass(MySingletonClass);
    DeclareSingleton(MySingletonClass);
public:
    /// constructor
    MySingletonClass();
    /// destructor
    virtual ~MySingletonClass();
    ...;
};

// implement the singleton class
ImplementClass(MyNamespace::MySingletonClass, 'MYSC', Core::RefCounted)
ImplementSingleton(MyNamespace::MySingletonClass);

//** Implements the Singleton constructor. */
MySingletonClass::MySingletonClass()
{
    ConstructSingleton;
}
```
The DeclareSingleton() and ImplementSingleton() macros are similar to the DeclareClass() and ImplementClass() macros. They add some static methods to the class (namely the Instance() and HasInstance() methods). The constructor and destructor of the class must contain a ConstructSingleton and DestructSingleton macros. ConstructSingleton initializes a private static singleton pointer and makes sure that no other instance of the class exists (otherwise, an assertion will be thrown). DestructSingleton invalidates the static singleton pointer.

Access to singletons is by default thread-local. This means that a singleton created in one thread of a Nebula3 application isn't accessible from another thread. This follows the "Parallel Nebulas" paradigm which simplifies multithreaded programming a lot. The idea behind "Parallel Nebulas" is, that a typical Nebula3 application contains of a few "fat threads" each running ideally on a separate CPU core. Fat threads implement for instance asynchronous IO, rendering, physics, and so on. Each of those fat threads initializes its own Nebula3 runtime, which just contains the minimal Nebula3 environment needed to perform the Fat Threads specific task. This basically eliminates the need for fine-grained synchronisation in almost all of the Nebula3 code and concentrates "thread-aware" code to a few well-defined code areas which deals with communication between fat threads. Another positive side effect of the "Parallel Nebulas" paradigm is, that a programmer typically doesn't have to care too much about running in a multithreaded environment. Most of the typical Nebula3 code looks just like normal singlethreaded code, yet can still run in its own fat thread.

Performance And Memory Footprint Considerations
One of the design goals of the Nebula3 Core Layer was to reduce the memory footprint of low level code to make the system better suited for small host platforms like handheld consoles (and a small memory footprint doesn't hurt on bigger platforms either). Here are some points how these goals are accomplished:

- The **RefCounted** class just adds 4 bytes per-instance data for the reference count member, Nebula2's nRoot class added >60 bytes overhead to each instance.
- The RTTI mechanism adds somewhere between 30 and 60 bytes overhead, but this is per-class, not per instance.
- A smart pointer is just 4 bytes, just like a raw pointer. The similar Nebula2 nRef class was 16 bytes per instance.
- Several householding structures are only allocated in debug mode, most notably the **RefCountedList**, which is used to detect refcounting leaks.

Here are some timings for creating a million **RefCounted** objects by the 3 different ways. These timings are on a notebool with Intel Pentium M running at 800 MHz:

- Create(): 0.29 seconds
- by FourCC: 0.65 seconds
- by class name: 1.45 seconds
# Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoreServer</td>
<td></td>
</tr>
<tr>
<td>ExitHandler</td>
<td></td>
</tr>
<tr>
<td>Factory</td>
<td></td>
</tr>
<tr>
<td>RefCounted</td>
<td></td>
</tr>
<tr>
<td>RefCountedList</td>
<td></td>
</tr>
<tr>
<td>Rtti</td>
<td></td>
</tr>
<tr>
<td>SysFunc</td>
<td></td>
</tr>
<tr>
<td>Singleton</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](http://www.doxygen.org/) at Fri Mar 26 15:21:42 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
CoreAnimation Namespace Reference
Detailed Description

The **CoreAnimation** subsystem contains the base functionality for higher level animation-related systems.

FIXME: continue...
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AnimClip</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AnimCurve</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AnimEvent</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AnimEventEmitter</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AnimKeyBuffer</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AnimResource</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AnimSampleBuffer</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AnimSampleMixInfo</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AnimUtil</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CurveType</strong></td>
<td></td>
</tr>
<tr>
<td><strong>InfinityType</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Nax3Header</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Nax2Header</strong></td>
<td></td>
</tr>
<tr>
<td><strong>SampleType</strong></td>
<td></td>
</tr>
<tr>
<td><strong>StreamAnimationLoader</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ManagedAnimation</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>AnimSampleJobFunc</code></td>
<td>(const JobFuncContext &amp;ctx)</td>
</tr>
<tr>
<td>void</td>
<td><code>AnimSampleMixJobFunc</code></td>
<td>(const JobFuncContext &amp;ctx)</td>
</tr>
<tr>
<td>void</td>
<td><code>AnimJobUtilSampleStep</code></td>
<td>(const <code>AnimCurve</code> *curves, int numCurves, const <code>float4</code> &amp;velocityScale, const <code>float4</code> *src0SamplePtr, <code>float4</code> *outSamplePtr, uchar *outSampleCounts)</td>
</tr>
<tr>
<td>void</td>
<td><code>AnimJobUtilSampleLinear</code></td>
<td>(const <code>AnimCurve</code> *curves, int numCurves, float sampleWeight, const <code>float4</code> &amp;velocityScale, const <code>float4</code> *src0SamplePtr, const <code>float4</code> *src1SamplePtr, <code>float4</code> *outSamplePtr, uchar *outSampleCounts)</td>
</tr>
<tr>
<td>void</td>
<td><code>AnimJobUtilMix</code></td>
<td>(const <code>AnimCurve</code> *curves, int numCurves, float mixWeight, const <code>float4</code> *src0SamplePtr, const <code>float4</code> *src1SamplePtr, const uchar *src0SampleCounts, const uchar *src1SampleCounts, <code>float4</code> *outSamplePtr, uchar *outSampleCounts)</td>
</tr>
</tbody>
</table>
Function Documentation

```c
void CoreAnimation::AnimSampleJobFunc (JobFuncContext ctx)

This job function only performs sampling, not mixing. This is usually called for the first anim job of a mixing chain.

void CoreAnimation::AnimSampleMixJobFunc (JobFuncContext ctx)

Performs both animation sampling, and mixing with another sample buffer in a single job.

void CoreAnimation::AnimJobUtilSampleStep (AnimCurve *curves, int numCurves, const float4 &velocityScale, const float4 *src0SamplePtr, float4 *outSamplePtr, uchar *outSampleCounts)

Sampler for "step" interpolation type.

void CoreAnimation::AnimJobUtilSampleLinear (AnimCurve *curves, int numCurves, float sampleWeight, const float4 &velocityScale, const float4 *src0SamplePtr, const float4 *src1SamplePtr, float4 *outSamplePtr, uchar *outSampleCounts)
```
Sampler for "linear" interpolation type.

```c
void CoreAnimation::AnimJobUtilMix(AnimCurve curves, const int numCurves, float mixWeight, const float4 *src0SamplePtr, const float4 *src1SamplePtr, const uchar *src0SampleCounts, const uchar *src1SampleCounts, float4 *outSamplePtr, uchar *outSampleCounts)
```

Mixes 2 source sample buffers into a destination sample buffer using a single lerp-value between 0.0 and 1.0. Mixing takes sample counts into consideration. A source sample count of 0 indicates, the this sample is not valid and the result is made of 100% of the other sample. If both source samples are valid, the result is blended from both source samples. This gives the expected results when an animation clip only manipulates parts of a character skeleton.

NOTE: the output data blocks may be identical with one of the input data blocks!
CoreGraphics Namespace Reference
Detailed Description

FIXME!
# Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapter</td>
</tr>
<tr>
<td>AdapterInfo</td>
</tr>
<tr>
<td>AntiAliasQuality</td>
</tr>
<tr>
<td>BatchType</td>
</tr>
<tr>
<td>DisplayDevice</td>
</tr>
<tr>
<td>DisplayEvent</td>
</tr>
<tr>
<td>DisplayEventHandler</td>
</tr>
<tr>
<td>DisplayMode</td>
</tr>
<tr>
<td>ImageFileFormat</td>
</tr>
<tr>
<td>IndexBuffer</td>
</tr>
<tr>
<td>IndexType</td>
</tr>
<tr>
<td>MemoryIndexBufferLoader</td>
</tr>
<tr>
<td>MemoryMeshLoader</td>
</tr>
<tr>
<td>MemoryVertexBufferLoader</td>
</tr>
<tr>
<td>Mesh</td>
</tr>
<tr>
<td>MousePointer</td>
</tr>
<tr>
<td>MouseRenderDevice</td>
</tr>
<tr>
<td>MultipleRenderTarget</td>
</tr>
<tr>
<td>PixelFormat</td>
</tr>
<tr>
<td>PrimitiveGroup</td>
</tr>
<tr>
<td>PrimitiveTopology</td>
</tr>
<tr>
<td>RenderDevice</td>
</tr>
<tr>
<td>RenderEvent</td>
</tr>
<tr>
<td>RenderEventHandler</td>
</tr>
<tr>
<td>RenderShape</td>
</tr>
<tr>
<td>RenderTarget</td>
</tr>
<tr>
<td>Shader</td>
</tr>
<tr>
<td>ShaderFeature</td>
</tr>
<tr>
<td>ShaderInstance</td>
</tr>
<tr>
<td>Class</td>
</tr>
<tr>
<td>-----------------------</td>
</tr>
<tr>
<td>ShaderServer</td>
</tr>
<tr>
<td>ShaderVariable</td>
</tr>
<tr>
<td>ShaderVariableInstance</td>
</tr>
<tr>
<td>ShaderVariation</td>
</tr>
<tr>
<td>ShapeRenderer</td>
</tr>
<tr>
<td>StreamMeshLoader</td>
</tr>
<tr>
<td>StreamShaderLoader</td>
</tr>
<tr>
<td>StreamTextureLoader</td>
</tr>
<tr>
<td>StreamTextureSaver</td>
</tr>
<tr>
<td>TextElement</td>
</tr>
<tr>
<td>TextRenderer</td>
</tr>
<tr>
<td>Texture</td>
</tr>
<tr>
<td>ThreadSafeDisplayEventHandler</td>
</tr>
<tr>
<td>ThreadSafeRenderEventHandler</td>
</tr>
<tr>
<td>TransformDevice</td>
</tr>
<tr>
<td>VertexBuffer</td>
</tr>
<tr>
<td>VertexComponent</td>
</tr>
<tr>
<td>VertexLayout</td>
</tr>
<tr>
<td>VertexLayoutServer</td>
</tr>
</tbody>
</table>
Debug Namespace Reference
Detailed Description

The Nebula3 Debug Subsystem

The Nebula3 Debug subsystem contains lowlevel classes which aid in runtime- and postmortem-debugging.
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>CorePageHandler</td>
</tr>
<tr>
<td>DebugCounter</td>
</tr>
<tr>
<td>DebugFloat</td>
</tr>
<tr>
<td>DebugHandler</td>
</tr>
<tr>
<td>DebugInterface</td>
</tr>
<tr>
<td>DebugPageHandler</td>
</tr>
<tr>
<td>DebugServer</td>
</tr>
<tr>
<td>DebugTimer</td>
</tr>
<tr>
<td>MiniDump</td>
</tr>
<tr>
<td>HelloWorldRequestHandler</td>
</tr>
<tr>
<td>SvgTestPageHandler</td>
</tr>
<tr>
<td>ConsolePageHandler</td>
</tr>
<tr>
<td>IoPageHandler</td>
</tr>
<tr>
<td>MemoryPageHandler</td>
</tr>
<tr>
<td>ThreadPageHandler</td>
</tr>
<tr>
<td>DisplayPageHandler</td>
</tr>
<tr>
<td>MeshPageHandler</td>
</tr>
<tr>
<td>ShaderPageHandler</td>
</tr>
<tr>
<td>TexturePageHandler</td>
</tr>
<tr>
<td>DebugGraphicsHandler</td>
</tr>
<tr>
<td>DebugShapeRenderer</td>
</tr>
<tr>
<td>DebugTextRenderer</td>
</tr>
<tr>
<td>GraphicsPageHandler</td>
</tr>
<tr>
<td>StreamingTexturePageHandler</td>
</tr>
<tr>
<td>ObjectInspectorHandler</td>
</tr>
</tbody>
</table>
Direct3D9 Namespace Reference
Detailed Description

FIXME!
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3D9DisplayDevice</td>
</tr>
<tr>
<td>D3D9RenderDevice</td>
</tr>
<tr>
<td>D3D9RenderTarget</td>
</tr>
<tr>
<td>D3D9Shader</td>
</tr>
<tr>
<td>D3D9ShaderInstance</td>
</tr>
<tr>
<td>D3D9ShaderServer</td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
</tr>
<tr>
<td>D3D9ShaderVariation</td>
</tr>
<tr>
<td>D3D9StreamShaderLoader</td>
</tr>
<tr>
<td>D3D9StreamTextureLoader</td>
</tr>
<tr>
<td>D3D9TextRenderer</td>
</tr>
<tr>
<td>D3D9Texture</td>
</tr>
<tr>
<td>D3D9ParticleRenderer</td>
</tr>
<tr>
<td>D3D9ParticleSystemInstance</td>
</tr>
</tbody>
</table>
Frame Namespace Reference
Detailed Description

FIXME!
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>FrameBatch</td>
</tr>
<tr>
<td>FramePass</td>
</tr>
<tr>
<td>FramePassBase</td>
</tr>
<tr>
<td>FramePostEffect</td>
</tr>
<tr>
<td>FrameServer</td>
</tr>
<tr>
<td>FrameShader</td>
</tr>
<tr>
<td>FrameShaderLoader</td>
</tr>
<tr>
<td>LightingMode</td>
</tr>
<tr>
<td>SortingMode</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:43 2010
Game Namespace Reference
Detailed Description

The **Game** namespace contains all base classes for game logic such as entities, property and manager.
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity</strong></td>
</tr>
<tr>
<td><strong>FeatureUnit</strong></td>
</tr>
<tr>
<td><strong>GameServer</strong></td>
</tr>
<tr>
<td><strong>Manager</strong></td>
</tr>
<tr>
<td><strong>Property</strong></td>
</tr>
<tr>
<td><strong>BaseGameFeatureUnit</strong></td>
</tr>
<tr>
<td><strong>PhysicsFeatureUnit</strong></td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
Graphics Namespace Reference
Detailed Description

This file was generated with Nebula3's idlc compiler tool. DO NOT EDIT
Data Structures

class AbstractLightEntity
class CameraEntity
class Display
class DisplaySettings
class GlobalLightEntity
class GraphicsEntity
class GraphicsHandler
class GraphicsInterface
class GraphicsServer
class ModelEntity
class MouseRenderer
class PointLightEntity
class SpotLightEntity
class View
class GraphicsEntityType
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
Detailed Description

The GraphicsFeature namespace contains properties and messages for using renderable graphics with the entity system of the game application.
# Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>LightFlickerUtil</td>
</tr>
<tr>
<td>SegmentedGfxUtil</td>
</tr>
<tr>
<td>AttachmentManager</td>
</tr>
<tr>
<td>ActorGraphicsProperty</td>
</tr>
<tr>
<td>AnimationControlProperty</td>
</tr>
<tr>
<td>CameraProperty</td>
</tr>
<tr>
<td>ChaseCameraProperty</td>
</tr>
<tr>
<td>GraphicsProperty</td>
</tr>
<tr>
<td>MayaCameraProperty</td>
</tr>
<tr>
<td>GraphicsFeatureUnitUnit</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](http://www.doxygen.org) at Fri Mar 26 15:21:45 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
Http Namespace Reference
Detailed Description

The Nebula3 Http Subsystem

The **Http** subsystem offers a complete set of HTTP client- and server-classes. The **HttpServer** class implements simple but complete builtin HTTP server which can communicate with web browsers. The **HttpServer** is mainly used to expose runtime debugging data, but it can also be used for general HTTP communication with the Nebula3 application. A **HtmlPageWriter** class exists to simplify writing HTML-formatted pages to an **IO::Stream**. **HttpRequestReader** is a **IO::StreamReader** which can decode an HTTP request from a web browser. The **HttpResponseWriter** is a **IO::StreamWriter** which encodes a HTTP response with attached content (usually a HTML page or an image) into an **IO::Stream**.

HttpRequestHandler's process HTTP requests and create a content stream which is sent back to client web browsers. The **HttpServer** is extended with new functionality by deriving new subclasses from **HttpRequestHandler** and adding instances to the **HttpServer**.

To connect to a running Nebula3 application on the same machine, open a web browser and navigate to the following address:

http://127.0.0.1:2100

This should open the applications main page with links to specialized pages implemented by custom HttpRequestHandlers.
### Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultHttpRequestHandler</td>
<td></td>
</tr>
<tr>
<td>HtmlElement</td>
<td></td>
</tr>
<tr>
<td>HtmlPageWriter</td>
<td></td>
</tr>
<tr>
<td>HttpMethod</td>
<td></td>
</tr>
<tr>
<td>HttpRequest</td>
<td></td>
</tr>
<tr>
<td>HttpRequestHandler</td>
<td></td>
</tr>
<tr>
<td>HttpRequestReader</td>
<td></td>
</tr>
<tr>
<td>HttpRequestWriter</td>
<td></td>
</tr>
<tr>
<td>HttpResponseReader</td>
<td></td>
</tr>
<tr>
<td>HttpResponseWriter</td>
<td></td>
</tr>
<tr>
<td>HttpServer</td>
<td></td>
</tr>
<tr>
<td>HttpServerProxy</td>
<td></td>
</tr>
<tr>
<td>HttpStatus</td>
<td></td>
</tr>
<tr>
<td>SvgLineChartWriter</td>
<td></td>
</tr>
<tr>
<td>SvgPageWriter</td>
<td></td>
</tr>
<tr>
<td>HttpClient</td>
<td></td>
</tr>
<tr>
<td>HttpClientRegistry</td>
<td></td>
</tr>
<tr>
<td>Http NzStream</td>
<td></td>
</tr>
<tr>
<td>HttpStream</td>
<td></td>
</tr>
</tbody>
</table>
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
Input Namespace Reference
Detailed Description

FIXME!
### Data Structures

- `class GamePad`
- `class InputEvent`
- `class InputHandler`
- `class InputPriority`
- `class InputServer`
- `class Key`
- `class Keyboard`
- `class Mouse`
- `class MouseButton`

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:46 2010
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
Interface Namespace Reference
Detailed Description

**Interface** objects offer asynchronous access to important Nebula3 subsystems, like IO, Audio and Rendering. They are the key mechanism in Nebula3 to make efficient use of multicore CPUs.

**Interface** objects are thread-global singletons which encapsulate an entire Nebula3 subsystem. In fact each **Interface** object initializes a minimal Nebula3 runtime which runs in a separate thread. This works great because all Nebula3 singletons are thread-local (the only exception are **Interface** objects which are singletons across threads). So code running in a parallel thread will only see the Nebula3 server objects which have been created in this thread. This parallel approach simplifies the implementation of Nebula3 classes, since only the code which handles communication across threads has to be thread safe.

Communication with **Interface** objects happens through normal Message objects. To instruct an **Interface** object to execute a task in parallel to the main thread, just create a Message object, initialize it and send it through to the Interface's Send() method. The Send() method will return immediately and the calling thread is free to do other stuff while the task is executing in the **Interface** object's worker thread. If a result is desired, the calling thread can either poll or wait for completion of the original message.
Data Structures

class InterfaceBase
class InterfaceHandlerBase

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:46 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
IO Namespace Reference
Detailed Description

The Nebula3 IO Subsystem

The Nebula3 IO subsystem is a huge step forward from the Nebula1 and Nebula2 IO systems. The main design goals of the new IO subsystem are:

- use more standard mechanisms, like URIs to identify resource locations, and MIME types to identify data formats
- a flexible stream model, it shouldn't matter whether data comes from a file, from a memory buffer, an HTTP connection or somewhere else
- reading and writing data from and to a stream in different data formats should be more orthogonal, for instance it shouldn't matter if XML-formatted data is read from memory, from a file, from a network connection or from anywhere else
- extensibility, new stream and reader/writer classes can be registered with the IO subsystem at runtime
- portability without performance compromises, the entire IO subsystem must be able to use platform-specific IO functions under the hood instead of relying to CLib functions like fopen() for portability, which may come with an additional performance or memory overhead compared to the platform specific IO functions

The main concepts of the Nebula3 IO subsystem are:

- A central IO::Console object for all text input and output with attachable console handlers. It is guaranteed that all Nebula3 text output goes through the console as the one centralized in/out channel. Specialized console handlers can be used to handle text output in a special way (for instance writing the output to stdout, an ingame console, a log file or a network connection).
- Assigns as path aliases. The general functionality is the same as in Nebula1 and Nebula2, or the original AmigaOS assigns which inspired Nebula's assign mechanism. A new feature of Nebula3 assigns is that they can be aliases for complete URI's. For
instance, the assign "textures:" could be defined as "http://www.radonlabs.de/textures", so that the shortcut resource path "textures:mytexture.dds" would be resolve to the absolute location "http://www.radonlabs.de/textures/mytexture.dds"

- **Streams** as basic in/out data channel. Streams are the replacement for Nebula2's nFile objects. Streams offer the same basic API with Open()/Close()/Read()/Write(), but may hide completely different transport or storage channels behind their common interface. Examples of stream classes are **IO::FileStream**, **IO::MemoryStream**, or **Net::HttpStream**
- **Stream readers and writers** are attached to streams and generally implement easy-to-use interfaces to read and write different data formats. For instance one could attach an **IO::XmlReader** to a **IO::FileStream** to read XML-formatted data from a filesystem file, or attach it to an **IO::HttpStream** to read XML-formatted data from a HTTP connection.

A good example to show the power of the Nebula3 in/out system is the following code fragment:

```cpp
IO::FileServer::Instance()->CopyFile("http://www.radonlabs.de/index.html")
```

This single line of code copies a file from a HTTP server into the current user's temp directory. With only a few lines more you could create a stream object pointing to a HTML file on a HTTP server, attach an XML reader to the stream, and parse the content of the HTML file without any intermediate storage file.

**Nebula3 Standard Assigns**

Nebula3 initializes the following standard assigns:

- **home**: Points to the application directory, in a German Windows installation, this is usually somewhere under "C:/Programme". Nebula3 applications should treat the home: location as a read only directory so that the user doesn't need administrator rights to run the application.
- **user**: This points to the currently logged in user directory for this
Nebula3 application. In a German Windows installation, this is somewhere under "C:/Eigene Dateien/[username]". Nebula3 will automatically create a local directory in the user directory to prevent different applications to overwrite their data. It is generally safe to write data to the user directory. This is the place where configuration and save game data should be written, or any other data which should persist between application invokations.

- **temp**: This assign points to the current user’s temp directory. This directory is generally writable. It should not be assumed that data in temp: survives until the next application start.
- **bin**: This points to the directory of the application’s executable file. This may or may not be identical with the home: directory. The bin: assign should be treated as read-only.

Custom assigns may be defined at runtime by the application. Often this is used to define abstract path to resources like textures, sound data, and so on. That way the locations of those resources can be easily changes by setting a single assign instead of fixing all the resource paths. A nice side effect of assigns is that a path with assigns is often much shorter then an "absolute" path resulting in a smaller memory footprint.

**Nebula3 URIs**

Resource locations are generally defined through standard URIs in Nebula3. URIs may consist of the following parts, some of them optional:

- a scheme, for instance "http:“, "file:“, etc... Nebula3 doesn’t define any hardcoded schemes, instead, schemes are bound to stream classes by registering them with the IO::StreamServer singleton
- an optional user info field, often this is a login name and a password to authenticate with a remote FTP or HTTP host
- a hostname, like "www.radonlabs.de"
- an optional portname following the hostname
- a local path, pointing to a resource on the host
- an optional fragment, which often points to a location inside the resource
- an optional query part, which often contains arguments for a PHP
script or some similar dynamic response mechanism

The class `IO::URI` is used to pass URIs around and to crack URI strings into its various components. It should be noted however, that an URI object has a bigger memory footprint compared to storing the URI in a simple string. So sometimes it may be better keep URIs around in strings and only use the `IO::URI` class to split the URI string into its parts.

Here are some examples for URI's:

<table>
<thead>
<tr>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>file://c:/temp/bla.txt</td>
</tr>
<tr>
<td>file://samba/temp/bla.txt</td>
</tr>
<tr>
<td><a href="http://www.radonlabs.de/index.html">http://www.radonlabs.de/index.html</a></td>
</tr>
<tr>
<td><a href="http://user:password@www.myserver.com:8080/index.html#main">http://user:password@www.myserver.com:8080/index.html#main</a></td>
</tr>
</tbody>
</table>

By using Nebula3 assigns you can simplify those complex pathnames a lot. To reference a file in the application directory you can write for instance `home:bla.txt` which would resolve to something like `file:///c:/programme/[myapp]/bla.txt`.

**Nebula3 Streams, Readers and Writers**

Streams provide a common interface for storing or transporting raw data. They replace the nFile class of Nebula2 with a much more general approach for storing, retrieving and transporting data. A stream object provides the traditional Open()/Close()/Read()/Write()/Seek() interface. Some stream classes provide memory mapping, so that data can be read or written by direct memory access. **Stream** objects use an **IO::URI** object to define their resource location. Usually, one URI scheme maps to a specific stream class. For instance the URI scheme "http:" usually maps to the Net::HttpStream class, while the scheme "file:" maps to the **IO::Filestream** class. This mapping is implemented by the StreamServer which constructs a matching stream object given an URI. A Nebula3 application is responsible to provide the mapping of URI scheme to stream classes using the StreamServer::Register() method. This is also the way how new stream classes and schemes
are registered with Nebula3.

Important stream classes in Nebula3 are for instance:

- **IO::FileStream**: provides access to the host's filesystem
- **IO::MemoryStream**: a dynamic memory buffer with a stream interface
- **IO::HttpStream**: provides a stream interface to files on a HTTP server

To read and write formatted stream data in a more flexible way then Nebula2, stream reader and stream writer classes have been introduced in Nebula3. **Stream** reader and writer classes provide a comfortable interface which is specialized on a specific data format. Here are some examples of stream readers and writers provided by Nebula3:

- **IO::BinaryReader/IOBinaryWriter**: read and write binary data
- **IO::TextReader/IOTextWriter**: read and write text and character data
- **IO::XmlReader/IOXmlWriter**: read and write XML formatted data
- **Messaging::MessageReader/MessagingMessageWriter**: Message serialization

Here's a simple example how to access a file on a HTTP server with a **XmlReader**:

```cpp
using namespace IO;

Ptr<Stream> stream = StreamServer::Instance()->CreateStream("http://www.radonlabs.de/index.html");
Ptr<XmlReader> xmlReader = XmlReader::Create();
xmlReader->SetStream(stream);
if (xmlReader->Open())
{
    // parse content here using the XmlReader interface
}
```

**The Nebula3 File Server**

The Nebula3 IO::FileServer class provides a singleton which offers
access to the hosts filesystem for global operations like defining
assigns, copying, deleting and checking for existence of files and
directories, listing directory contents, and so on.

Here's some sample code fragments for some of the more useful
FileServer methods:

```cpp
using namespace IO;
using namespace Util;

FileServer* fs = FileServer::Instance();

// check if a file or directory exists
bool fileExists = fs->FileExists("home:bla.txt");
bool dirExists = fs->DirectoryExists("temp:bla/blub");

// resolve a path with assigns into an absolute filesystem
// path, this is sometimes necessary to interface with
// 3rd party libraries which don't understand Nebula3 paths
dir
String absPath = fs->ResolveAssings("user:myapp/savegames");

// create a directory, note that all missing subdirectories will
// be created as well
fs->CreateDirectory("user:myapp/savegames");

// copy and delete files
fs->CopyFile("home:movie.mpg", "temp:movie.mpg");
fs->DeleteFile("temp:movie.mpg");

// list files in a directory matching a pattern
Array<String> files = fs->ListFiles("temp:", "*.txt");

// list all subdirectories in temp:
Array<String> dirs = fs->ListDirectories("temp:", "*");
```

The Nebula3 Console

[TODO]
# Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>Archive</td>
</tr>
<tr>
<td>class</td>
<td>ArchiveBase</td>
</tr>
<tr>
<td>class</td>
<td>ArchiveFileSystem</td>
</tr>
<tr>
<td>class</td>
<td>Assign</td>
</tr>
<tr>
<td>class</td>
<td>AssignRegistry</td>
</tr>
<tr>
<td>class</td>
<td>BinaryReader</td>
</tr>
<tr>
<td>class</td>
<td>BinaryWriter</td>
</tr>
<tr>
<td>class</td>
<td>BXmlReader</td>
</tr>
<tr>
<td>class</td>
<td>Console</td>
</tr>
<tr>
<td>class</td>
<td>ConsoleHandler</td>
</tr>
<tr>
<td>class</td>
<td>ExcelXmlReader</td>
</tr>
<tr>
<td>class</td>
<td>FileStream</td>
</tr>
<tr>
<td>class</td>
<td>FSWrapper</td>
</tr>
<tr>
<td>class</td>
<td>GameContentServer</td>
</tr>
<tr>
<td>class</td>
<td>HistoryConsoleHandler</td>
</tr>
<tr>
<td>class</td>
<td>IoInterfaceHandler</td>
</tr>
<tr>
<td>class</td>
<td>IoServer</td>
</tr>
<tr>
<td>class</td>
<td>LogFileConsoleHandler</td>
</tr>
<tr>
<td>class</td>
<td>MediaType</td>
</tr>
<tr>
<td>class</td>
<td>MemoryStream</td>
</tr>
<tr>
<td>class</td>
<td>SchemeRegistry</td>
</tr>
<tr>
<td>class</td>
<td>Stream</td>
</tr>
<tr>
<td>class</td>
<td>StreamReader</td>
</tr>
<tr>
<td>class</td>
<td>StreamWriter</td>
</tr>
<tr>
<td>class</td>
<td>TextReader</td>
</tr>
<tr>
<td>class</td>
<td>TextWriter</td>
</tr>
<tr>
<td>class</td>
<td>URI</td>
</tr>
<tr>
<td>class</td>
<td>BXmlILoaderUtil</td>
</tr>
<tr>
<td>class</td>
<td>XmlReader</td>
</tr>
<tr>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>XmlWriter</td>
<td></td>
</tr>
<tr>
<td>ZipArchive</td>
<td></td>
</tr>
<tr>
<td>ZipDirEntry</td>
<td></td>
</tr>
<tr>
<td>ZipFileEntry</td>
<td></td>
</tr>
<tr>
<td>ZipFileStream</td>
<td></td>
</tr>
<tr>
<td>ZipFileSystem</td>
<td></td>
</tr>
<tr>
<td>FileTime</td>
<td></td>
</tr>
<tr>
<td>Interface</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:46 2010
Detailed Description

The **Legacy** namespace contains various classes which support Nebula2 backward compatibility (mainly support for Nebula2 file formats). To compile Nebula3 without support for Nebula2 legacy support, set the `__NEBULA_LEGACY_SUPPORT__` define in in `/foundation/core/config.h` to (0).
Data Structures

class Nvx2StreamReader

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:47 2010
Lighting Namespace Reference
Detailed Description

class Lighting::ShadowServerBase

The ShadowServer setups and controls the global aspects of the dynamic shadow system.

(C) 2007 Radon Labs GmbH
### Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LightServerBase</code></td>
<td></td>
</tr>
<tr>
<td><code>InternalAbstractLightEntity</code></td>
<td></td>
</tr>
<tr>
<td><code>InternalGlobalLightEntity</code></td>
<td></td>
</tr>
<tr>
<td><code>InternalPointLightEntity</code></td>
<td></td>
</tr>
<tr>
<td><code>InternalSpotLightEntity</code></td>
<td></td>
</tr>
<tr>
<td><code>LightPrePassServer</code></td>
<td></td>
</tr>
<tr>
<td><code>LightServer</code></td>
<td></td>
</tr>
<tr>
<td><code>LightType</code></td>
<td></td>
</tr>
<tr>
<td><code>PSSMUtil</code></td>
<td></td>
</tr>
<tr>
<td><code>ShadowServer</code></td>
<td></td>
</tr>
<tr>
<td><code>SM30LightServer</code></td>
<td></td>
</tr>
<tr>
<td><code>SM30ShadowServer</code></td>
<td></td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
Loader Namespace Reference
Detailed Description

The Loader namespace contains all classes for loading levels from a database.
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
Math Namespace Reference
**Detailed Description**

Nebula3’s **Math** subsystem offers a standalone library of C++ math classes. Nothing unusual about them ;)

FIXME: make math code less object oriented and more like HLSL for better performance and portability to Xbox360.
# Data Structures

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbox</td>
<td>class</td>
</tr>
<tr>
<td>ClipStatus</td>
<td>class</td>
</tr>
<tr>
<td>Extrapolator</td>
<td>class</td>
</tr>
<tr>
<td>float2</td>
<td>class</td>
</tr>
<tr>
<td>frustum</td>
<td>class</td>
</tr>
<tr>
<td>line</td>
<td>class</td>
</tr>
<tr>
<td>noise</td>
<td>class</td>
</tr>
<tr>
<td>polar</td>
<td>class</td>
</tr>
<tr>
<td>rectangle</td>
<td>class</td>
</tr>
<tr>
<td>sphere</td>
<td>class</td>
</tr>
<tr>
<td>transform44</td>
<td>class</td>
</tr>
<tr>
<td>float4</td>
<td>class</td>
</tr>
<tr>
<td>matrix44</td>
<td>class</td>
</tr>
<tr>
<td>plane</td>
<td>class</td>
</tr>
<tr>
<td>point</td>
<td>class</td>
</tr>
<tr>
<td>quaternion</td>
<td>class</td>
</tr>
<tr>
<td>vector</td>
<td>class</td>
</tr>
</tbody>
</table>
### Functions

| __forceinline bool | `n_fequal` (scalar f0, scalar f1, scalar tol) |
| __forceinline int  | `n_iclamp` (int val, int minVal, int maxVal) |
| __forceinline bool | `n_fless` (scalar f0, scalar f1, scalar tol) |
| __forceinline bool | `n_fgreater` (scalar f0, scalar f1, scalar tol) |
| __forceinline scalar| `n_clamp` (scalar val, scalar lower, scalar upper) |
| __forceinline scalar| `n_saturate` (scalar val) |
| __forceinline double| `n_saturate` (double val) |
| __forceinline scalar| `n_lerp` (scalar x, scalar y, scalar l) |
| __forceinline double| `n_lerp` (double x, double y, double l) |
| __forceinline scalar| `n_angulardistance` (scalar from, scalar to) |
| __forceinline bool | `n_isdenormal` (scalar s) |
| __forceinline float | `n_undenormalize` (scalar s) |
| __forceinline bool | `n_nearequal` (scalar a, scalar b, scalar epsilon) |
| __forceinline scalar| `n_rand` () |
| __forceinline scalar| `n_rand` (scalar min, scalar max) |
| __forceinline int  | `n_fchop` (scalar f) |
| __forceinline scalar| `n_modangle` (scalar a) |
| __forceinline scalar| `n_log2` (scalar f) |
| __forceinline int  | `n_frnd` (scalar f) |
| __forceinline scalar| `n_log` (scalar x) |
Function Documentation

__forceinline
bool    ( scalar f0,
Math::n_fequal
    scalar f1,
    scalar tol
 )

A fuzzy floating point equality check

__forceinline
int    ( int val,
Math::n_iclamp
    int minVal,
    int maxVal
 )

Integer clamping.

__forceinline
bool    ( scalar f0,
Math::n_fless
    scalar f1,
    scalar tol
 )

A fuzzy floating point less-than check.

__forceinline
bool    ( scalar f0,
Math::n_fgreater
    scalar f1,
    scalar tol
 )

A fuzzy floating point greater-than check.

__forceinline
scalar    ( scalar val,
Math::n_clamp
    scalar lower,
Clamp a value against lower and upper boundary.

```cpp
__forceinline
scalar Math::n_saturate(scalar val)
```

Saturate a value (clamps between 0.0f and 1.0f)

```cpp
__forceinline
double Math::n_saturate(double val)
```

Saturate a value (clamps between 0.0f and 1.0f)

```cpp
__forceinline
scalar Math::n_lerp(scalar x, scalar y, scalar l)
```

Linearly interpolate between 2 values: ret = x + l * (y - x)

```cpp
__forceinline
double Math::n_lerp(double x, double y, double l)
```

Linearly interpolate between 2 values: ret = x + l * (y - x)

```cpp
__forceinline scalar Math::n_angulardistance(scalar from, scalar to)
```

Get angular distance.

```cpp
__forceinline bool Math::n_isdenormal(scalar s)
```
Returns true if the input scalar is denormalized (DEN)

```cpp
__forceinline float
Math::n_undenormalize ( scalar s )
```

Returns 0 if scalar is denormal.

```cpp
__forceinline bool
Math::n_nearequal ( scalar a,
        scalar b,
        scalar epsilon
    )
```

test of nearly equal given a tolerance (epsilon)

```cpp
__forceinline
scalar ( )
Math::n_rand
```

Return a pseudo random number between 0 and 1.

```cpp
__forceinline
scalar ( scalar min,
        scalar max
    )
Math::n_rand
```

Return a pseudo random number between min and max.

```cpp
__forceinline
int ( scalar f )
Math::n_fchop
```

Chop float to int.

```cpp
__forceinline
scalar ( scalar a )
Math::n_modangle
```

Normalize an angular value into the range rad(0) to rad(360).
log2() function.

```cpp
__forceinline
int Math::n_frnd(scalar f)
```

**Round** float to integer.

```cpp
__forceinline
scalar Math::n_log(scalar x)
```

get logarithm of x
Memory Namespace Reference
Detailed Description

The Nebula3 Memory subsystem implements custom memory allocation mechanisms which provide higher performance and better debugging aids.
## Data Structures

<table>
<thead>
<tr>
<th>Type</th>
<th>Class/Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct</td>
<td><code>TotalMemoryStatus</code></td>
</tr>
<tr>
<td>class</td>
<td><code>PoolArrayAllocator</code></td>
</tr>
<tr>
<td>class</td>
<td><code>Heap</code></td>
</tr>
<tr>
<td>class</td>
<td><code>Memory</code></td>
</tr>
<tr>
<td>class</td>
<td><code>MemoryPool</code></td>
</tr>
</tbody>
</table>
Enumerations

```c
enum HeapType
enum HeapType
```
## Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void * Alloc (HeapType heapType, size_t size, size_t alignment=16)</code></td>
<td>allocate a chunk of memory</td>
</tr>
<tr>
<td><code>void * Realloc (HeapType heapType, void *ptr, size_t size)</code></td>
<td>re-allocate a chunk of memory</td>
</tr>
<tr>
<td><code>void Free (HeapType heapType, void *ptr)</code></td>
<td>free a chunk of memory</td>
</tr>
<tr>
<td><code>char * DuplicateCString (const char *from)</code></td>
<td>duplicate a C-string (obsolete)</td>
</tr>
<tr>
<td><code>bool IsOverlapping (const unsigned char *srcPtr, size_t srcSize, const unsigned char *dstPtr, size_t dstSize)</code></td>
<td>check if 2 memory regions are overlapping</td>
</tr>
<tr>
<td><code>TotalMemoryStatus GetTotalMemoryStatus ()</code></td>
<td></td>
</tr>
<tr>
<td><code>void Copy (const void *from, void *to, size_t numBytes)</code></td>
<td>copy a chunk of memory</td>
</tr>
<tr>
<td><code>void Clear (void *ptr, size_t numBytes)</code></td>
<td>overwrite a chunk of memory with zero</td>
</tr>
<tr>
<td><code>void Fill (void *ptr, size_t numBytes, unsigned char value)</code></td>
<td>fill memory with a specific byte</td>
</tr>
<tr>
<td><code>void SetupHeaps ()</code></td>
<td></td>
</tr>
<tr>
<td><code>const char * GetHeapTypeName (HeapType heapType)</code></td>
<td></td>
</tr>
<tr>
<td><code>__forceinline void CopyToGraphicsMemory (const void *from, void *to, size_t numBytes)</code></td>
<td></td>
</tr>
<tr>
<td><code>void * Alloc (HeapType heapType, size_t size)</code></td>
<td>allocate a chunk of memory</td>
</tr>
<tr>
<td><code>void DumpTotalMemoryStatus ()</code></td>
<td></td>
</tr>
<tr>
<td><code>__forceinline unsigned char *__HeapAlignPointerAndWritePadding16 (unsigned char *ptr)</code></td>
<td></td>
</tr>
<tr>
<td>__forceinline unsigned char *</td>
<td><code>__HeapUnalignPointer16</code> (unsigned char *ptr)</td>
</tr>
<tr>
<td>__forceinline LPVOID</td>
<td><code>__HeapAlloc16</code> (HANDLE hHeap, DWORD dwFlags, SIZE_T dwBytes)</td>
</tr>
<tr>
<td>__forceinline LPVOID</td>
<td><code>__HeapReAlloc16</code> (HANDLE hHeap, DWORD dwFlags, LPVOID lpMem, SIZE_T dwBytes)</td>
</tr>
<tr>
<td>__forceinline BOOL</td>
<td><code>__HeapFree16</code> (HANDLE hHeap, DWORD dwFlags, LPVOID lpMem)</td>
</tr>
<tr>
<td>__forceinline SIZE_T</td>
<td><code>__HeapSize16</code> (HANDLE hHeap, DWORD dwFlags, LPCVOID lpMem)</td>
</tr>
</tbody>
</table>
Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>malloc_zone_t *</td>
<td>Heaps [NumHeapTypes]</td>
<td></td>
</tr>
<tr>
<td>malloc_zone_t *</td>
<td>Heaps [NumHeapTypes]</td>
<td></td>
</tr>
<tr>
<td>HANDLE volatile</td>
<td>Heaps [NumHeapTypes]</td>
<td>{ NULL }</td>
</tr>
<tr>
<td>HANDLE volatile</td>
<td>Heaps [NumHeapTypes]</td>
<td></td>
</tr>
</tbody>
</table>
Enumeration Type Documentation

enum Memory::HeapType

Heap types are defined here. The main purpose for the different heap types is to decrease memory fragmentation and to improve cache usage by grouping similar data together. Platform ports may define platform-specific heap types, as long as only platform specific code uses those new heap types.
Function Documentation

```c
void* Memory::Alloc (HeapType heapType,
                     size_t size,
                     size_t alignment = 16)
```
allocate a chunk of memory

Allocate a block of memory from one of the global heaps.

```c
void* Memory::Realloc (HeapType heapType,
                       void* ptr,
                       size_t size)
```
re-allocate a chunk of memory

Re-Allocate a block of memory from one of the global heaps.

NOTE that this function may also be used to shrink a memory block!

```c
void Memory::Free (HeapType heapType,
                    void* ptr)
```
free a chunk of memory

Free a block of memory.

```c
char* Memory::DuplicateCString (const char* from)
```
duplicate a C-string (obsolete)

Duplicate a 0-terminated string, this method should no longer be used!
bool Memory::IsOverlapping(const unsigned char * srcPtr, size_t srcSize, const unsigned char * dstPtr, size_t dstSize)

check if 2 memory regions are overlapping

Test if 2 areas of memory areas are overlapping.

**TotalMemoryStatus**
Memory::GetTotalMemoryStatus()

Get the system's total memory status.

void Memory::Copy(const void * from, void * to, size_t numBytes)

copy a chunk of memory

Copy a chunk of memory (note the argument order is different from memcpy()!!)

void Memory::Clear(void * ptr, size_t numBytes)

overwrite a chunk of memory with zero

Overwrite a chunk of memory with 0's.
fill memory with a specific byte

Fill memory with a specific byte.

```cpp
void Memory::Memory::SetupHeaps()

Setup the global heaps.

const char *
Memory::Memory::GetHeapTypeName(HeapType heapType)

Returns a human readable name for a heap type.

__forceinline void
Memory::CopyToGraphicsMemory(const void *from, void *to, size_t numBytes)

Copy data from a system memory buffer to graphics resource memory. Some platforms may need special handling of this case.

```cpp
void* Memory::Alloc(HeapType heapType, size_t size)

allocate a chunk of memory

Allocate a block of memory from one of the global heaps.

```cpp
void Memory::DumpTotalMemoryStatus()

Dump detail memory status information.

```cpp
__forceinline unsigned char*
Memory::__HeapAlignPointerAndWritePadding16(unsigned char *ptr)

Helper function for Heap16 functions: aligns pointer to 16 byte and
writes padding mask to byte before returned pointer.

```c
__forceinline unsigned char*
Memory::__HeapUnalignPointer16 ( unsigned char * ptr )
```

Helper function for Heap16 functions: "un-aligns" pointer through the padding mask stored in the byte before the pointer.

```c
__forceinline LPVOID
Memory::__HeapAlloc16 ( HANDLE hHeap,
    DWORD dwFlags,
    SIZE_T dwBytes
 )
```

HeapAlloc replacement which always returns 16-byte aligned addresses.

NOTE: only works for 32 bit pointers!

```c
__forceinline LPVOID
Memory::__HeapReAlloc16 ( HANDLE hHeap,
    DWORD dwFlags,
    LPVOID lpMem,
    SIZE_T dwBytes
 )
```

HeapReAlloc replacement for 16-byte alignment.

NOTE: only works for 32 bit pointers!

```c
__forceinline BOOL
Memory::__HeapFree16 ( HANDLE hHeap,
    DWORD dwFlags,
    LPVOID lpMem
 )
```

HeapFree replacement which always returns 16-byte aligned addresses.

NOTE: only works for 32 bit pointers!

```c
__forceinline SIZE_T
Memory::__HeapSize16 ( HANDLE hHeap,
```

```c
```
DWORD  dwFlags,
LPCVOID lpMem
)

HeapSize replacement function.
Variable Documentation

malloc_zone_t* Memory::Heaps[NumHeapTypes]

Heap pointers are defined here. Call ValidateHeap() to check whether a heap already has been setup, and to setup the heap if not.

malloc_zone_t* Memory::Heaps[NumHeapTypes]

Heap pointers are defined here. Call ValidateHeap() to check whether a heap already has been setup, and to setup the heap if not.

HANDLE volatile Memory::Heaps[NumHeapTypes] = { NULL }

Heap pointers are defined here. Call ValidateHeap() to check whether a heap already has been setup, and to setup the heap if not.

HANDLE volatile Memory::Heaps[NumHeapTypes]

Heap pointers are defined here. Call ValidateHeap() to check whether a heap already has been setup, and to setup the heap if not.
Messaging Namespace Reference
Detailed Description

FIXME!
# Data Structures

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsyncPort</td>
<td></td>
</tr>
<tr>
<td>BatchMessage</td>
<td></td>
</tr>
<tr>
<td>BlockingHandlerThread</td>
<td></td>
</tr>
<tr>
<td>DelegateTable</td>
<td></td>
</tr>
<tr>
<td>Dispatcher</td>
<td></td>
</tr>
<tr>
<td>Handler</td>
<td></td>
</tr>
<tr>
<td>HandlerThreadBase</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td></td>
</tr>
<tr>
<td>MessageReader</td>
<td></td>
</tr>
<tr>
<td>MessageWriter</td>
<td></td>
</tr>
<tr>
<td>Port</td>
<td></td>
</tr>
<tr>
<td>RunThroughHandlerThread</td>
<td></td>
</tr>
<tr>
<td>StaticMessageHandler</td>
<td></td>
</tr>
</tbody>
</table>
## Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>__Dispatcher (InternalCameraEntity)</td>
</tr>
<tr>
<td>__Dispatcher (InternalGraphicsEntity)</td>
</tr>
<tr>
<td>__Handler (InternalModelEntity, AnimIsClipPlaying)</td>
</tr>
<tr>
<td>__Handler (InternalModelEntity, BaseAnimEventMessage)</td>
</tr>
<tr>
<td>__Dispatcher (InternalModelEntity)</td>
</tr>
</tbody>
</table>
Function Documentation

Messaging::__Dispatcher ( InternalCameraEntity )

Dispatcher method (must be positioned after the handler methods to prevent automatic instantiation).

Messaging::__Dispatcher ( InternalGraphicsEntity )

Dispatcher method (must be positioned after the handler methods to prevent automatic instantiation).

Messaging::__Handler ( InternalModelEntity, AnimIsClipPlaying )

FIXME: this method doesn't really fit with the Animation System's philosophy...

Messaging::__Handler ( InternalModelEntity, BaseAnimEventMessage )

Special case, AnimEventMessages must be forwarded to the AnimEventServer, but only when the ModelEntity has loaded its resources (the character must be valid).

Messaging::__Dispatcher ( InternalModelEntity )

Dispatcher method (must be positioned after the handler methods to prevent automatic instantiation).
Models Namespace Reference
Detailed Description

FIXME!
Data Structures

- class ManagedModel
- class Model
- class ModelInstance
- class ModelNode
- class ModelNodeInstance
- class ModelNodeType
- class ModelServer
- class ShapeNode
- class ShapeNodeInstance
- class StateNode
- class StateNodeInstance
- class TransformNode
- class TransformNodeInstance
- class StreamModelLoader
- class VisResolveContainer
- class VisResolver
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
Detailed Description

The Nebula3 Net subsystem offers simple client/server-style communication using the TCP protocol over LAN or internet connections. It is not intended for highlevel game-oriented communication with lobbies, session management and synchronisation of player data. This will be provided in higher level Nebula3 networking subsystems.
Working with IP addresses

An **IpAddress** object identifies a communication endpoint by host name or tcp/ip address and a port number. **IpAddress** objects can be created in a number of ways:

```
// from TCP/IP address and port number:
IpAddress ipAddr("192.168.0.2", 1234);

// from host name and port number:
IpAddress ipAddr("www.radonlabs.de", 1234);

// from the local host (127.0.0.1) and port number:
IpAddress ipAddr("localhost", 1234);

// from the "any" address (0.0.0.0) and a port number:
IpAddress ipAddr("any", 1234);

// from the broadcast address (255.255.255.255) and a port number:
IpAddress ipAddr("broadcast", 1234);

// from the host's first valid network adapter's address and a port
IpAddress ipAddr("self", 1234);

// from the host's first valid network adapter connected to the int
IpAddress ipAddr("insetself", 1234);

// from an URI which defines a host name and a port number:
IpAddress ipAddr(IO::URI("http://www.radonlabs.de:2100"));
```

An **IpAddress** object can be used to lookup a TCP/IP address from a host name:

```
IpAddress ipAddr("www.radonlabs.de", 0);
String numericalAddr = ipAddr.GetHostAddr();
```
Setting Up A Client/Server System

The **Net** subsystem provides an easy-to-use TCP-based client/server system implemented in the classes **TcpServer** and **TcpClient** using the TCP protocol. Any number of TcpClients can be served by a single **TcpServer** simultaneously.

Setting up a server is done like this:

```cpp
using namespace Net;

Ptr<TcpServer> tcpServer = TcpServer::Create();
tcpServer->SetAddress(IpAddress("any", 2352));
if (tcpServer->Open())
{
    // TcpServer successfully opened
}
```

This will setup the server to listen on port 2352 for incoming client connection requests.

To communicate with the **TcpServer**, a **TcpClient** object needs to be setup on the client side:

```cpp
using namespace Net;

Ptr<TcpClient> tcpClient = TcpClient::Create();
tcpClient->SetBlocking(false);
tcpClient->SetAddress(IpAddress("localhost", 2352));
TcpClient::Result res = tcpClient->Connect();
```

This assumes that the server is running on the same machine as the client (since the client connects to "localhost").

In a non-blocking scenario as above, the Connect() method will either return with TcpClient::Success (which means the connection is established), or more likely with TcpClient::Connecting, in this case the connection hasn't been established yet, and the application needs to
continue calling the Connect() method. In the case of an connection error, the return code TcpClient::Error will be returned.

In a blocking scenario the Connect() method will not return until either the connection has been established (result would be TcpClient::Success) or an error occurred (TcpClient::Error).

**Note:**
An interactive application should never block during network communication and instead should provide continuous feedback to the user what's going on.

Once a connection has been established, a **TcpClientConnection** object will be created on the server side for each connected client. The **TcpClientConnection** represents the client on the server and is used to receive data from the client and to send responses back to the client.

For sending and receiving data, **IO::Stream** objects are used. By attaching **IO::StreamReader** and **IO::StreamWriter** objects to the communication streams it is very easy to encode and decode data from the stream.

**Note:**
Send-data is not sent immediately, instead the data will accumulate in the send stream until the Send() method is called.

To send some text data from a client to its server, obtain a pointer to the send stream, write data to it and call the Send() method:

```cpp
using namespace Net;
using namespace IO;

// obtain pointer to client's send stream and attach a TextWriter
const Ptr<Stream>& sendStream = tcpClient->GetSendStream();
Ptr<TextWriter> textWriter = TextWriter::Create();
textWriter->SetStream(sendStream);
textWriter->Open();
textWriter->WriteString("Hello Server");
textWriter->Close();
```
// send off the data to the server
if (this->tcpClient->Send())
{
    // data has been sent
}

To receive client data on the server side, the application needs to poll for `TcpClientConnection` which contain data from clients frequently (e.g. once per frame). More then one `TcpClientConnection` may be waiting for processing, thus the processing loop should look like this:

```cpp
using namespace Util;
using namespace IO;
using namespace Net;

// get array of client connections which received data since the last
Array<Ptr<TcpClientConnection>> recvConns = tcpServer->Recv();
IndexT i;
for (i = 0; i < recvConns.Size(); i++)
{
    // get receive stream from current connection, attach a text reader
    Ptr<TextReader> textReader = TextReader::Create();
    textReader->SetStream(recvConns[i]->GetRecvStream());
    textReader->Open();
    String str = textReader->ReadString();
    textReader->Close();

    // process received string and send response back to client
    // create a TextWriter and attach it to the send stream of the
    Ptr<TextWriter> textWriter = TextWriter::Create();
    textWriter->SetStream(recvConns[i]->GetSendStream());
    textWriter->Open();
    textWriter->WriteString("Hello Client");
    textWriter->Close();

    // finally send the response back to the client
    recvConns[i]->Send();
}
```

To get server responses on the client side, call the `TcpClient::Recv()` method which will block until data arrives (in blocking mode), or come back immediately (in non-blocking mode) and return true when data from the server is available:
using namespace Net;
using namespace IO;

// check if data is available from the server
if (tcpClient->Recv())
{
    // yep, data is available, get the recv stream and read the data
    const Ptr<Stream>& recvStream = tcpClient->GetRecvStream();
    Ptr<TextReader> textReader = TextReader::Create();
    textReader->SetStream(recvStream);
    textReader->Open();
    String responseString = textReader->ReadString();
    n_printf("The server said: %s\n", responseString.AsCharPtr());
    textReader->Close();
}

A client should also check whether the connection is still up by calling the IsConnected() method. If the connection has been dropped for some reason, this method will return false.

Note: 
TcpServer and TcpClient do not implement an underlying communication protocol which enables them to work with "foreign" clients and servers (for instance, a TcpServer could work with standard web browsers as client, and a TcpClient class could communicate with a standard HTTP server).

For real world scenarios, an application should implement its own robust communication protocol which at least encodes the length of the payload data. If the payload is bigger then some maximum packet size, data may be sent in several packets, and thus may arrive in several packets at the client. The client should decode the length of the payload from the message header to decide whether the received data represents a complete message, or whether more data needs to be received until message is complete.
Byte Order Issues

Servers and clients may run on CPUs with different byte order. If binary data is sent over a network connection, the data must be converted into a "network byte order" which both clients agree on. Nebula3 offers automatic byte order conversion in the **IO::BinaryReader** and **IO::BinaryWriter** classes. Simply call the following methods before reading from or writing to a network communication stream:

```cpp
binaryReader->SetStreamByteOrder(System::ByteOrder::Network);
binaryWriter->SetStreamByteOrder(System::ByteOrder::Network);
```
The Socket Class

The **Net** subsystem provides a **Socket** class which wraps the traditional socket functions into a C++ interface. Usually an application doesn't use **Socket** class directly and instead uses higher level networking classes like **TcpServer**. But if that's not possible for some reason the **Socket** class is much more convenient then working directly with socket functions.
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>DebugMessage</td>
</tr>
<tr>
<td>DebugPacket</td>
</tr>
<tr>
<td>MessageClient</td>
</tr>
<tr>
<td>MessageClientConnection</td>
</tr>
<tr>
<td>Socket</td>
</tr>
<tr>
<td>StdTcpClient</td>
</tr>
<tr>
<td>StdTcpClientConnection</td>
</tr>
<tr>
<td>StdTcpServer</td>
</tr>
<tr>
<td>TcpClient</td>
</tr>
<tr>
<td>TcpClientConnection</td>
</tr>
<tr>
<td>TcpMessageCodec</td>
</tr>
<tr>
<td>TcpServer</td>
</tr>
<tr>
<td>IpAddress</td>
</tr>
</tbody>
</table>
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
PhysicsFeature Namespace Reference
Detailed Description

The **PhysicsFeature** namespace contains properties and messages for using collision and physics with the entity system of the game application.
# Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td><code>ActorPhysicsProperty</code></td>
</tr>
<tr>
<td>class</td>
<td><code>EnvironmentCollideProperty</code></td>
</tr>
<tr>
<td>class</td>
<td><code>MouseGripperProperty</code></td>
</tr>
<tr>
<td>class</td>
<td><code>PhysicsProperty</code></td>
</tr>
<tr>
<td>class</td>
<td><code>TriggerProperty</code></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:49 2010
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
RenderUtil Namespace Reference
Detailed Description

FIXME!
Data Structures

- class DrawFullScreenQuad
- class MayaCameraUtil
- class MouseRayUtil
- class NodeLookupUtil
Resources Namespace Reference
Detailed Description

FIXME!
## Data Structures

<table>
<thead>
<tr>
<th>Class/Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td><strong>D3D9TextureStreamer</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>ManagedMesh</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>ManagedResource</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>ManagedTexture</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>Resource</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>ResourceDictionary</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>ResourceLoader</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>ResourceManager</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>ResourceMapper</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>ResourceSaver</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>SimpleResourceMapper</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>LoadingResource</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>PoolLoadingResource</strong></td>
</tr>
<tr>
<td>struct</td>
<td><strong>PoolSetupInfo</strong></td>
</tr>
</tbody>
</table>

*helper class for pool generation* [More...](#)

<table>
<thead>
<tr>
<th>Class/Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td><strong>PoolResourceMapper</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>ResourceScheduler</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>TexturePoolMapperScheduler</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>TextureStreamer</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>StreamResourceLoader</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>ResourceId</strong></td>
</tr>
</tbody>
</table>
• Main Page
•Namespaces
•Data Structures
•Files
•Related Pages

•Namespace List
•Namespace Members
ScriptFeature Namespace Reference
Detailed Description

This file was generated with Nebula3's idlc compiler tool. DO NOT EDIT
Data Structures

class ScriptFeatureUnit
Detailed Description

This file was generated with Nebula3's idlc compiler tool. DO NOT EDIT
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>StateGraphicsProperty</td>
</tr>
<tr>
<td>StateInfo</td>
</tr>
<tr>
<td>StateProperty</td>
</tr>
<tr>
<td>StateObjectFeatureUnit</td>
</tr>
</tbody>
</table>
Detailed Description

The Nebula3 **System** subsystem offers close-to-the-metal classes which expose information about or manipulate system-level aspects of the host platform, for instance information about the CPU, byte order or available hardware capabilities. Some classes offer access to platform-specific features, like the Windows registry.
Data Structures

class ByteOrder
class SystemInfo
class Win32Environment
class Win32Registry
class AppEntry
class Cpu
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
Threading Namespace Reference
Detailed Description

FIXME!
**Data Structures**

class **CriticalSection**
class **Event**
class **Interlocked**
class **ObjectRef**
class **SafeFlag**
class **SafePriorityQueue**
class **SafeQueue**
class **Thread**
class **ThreadBarrier**
class **Barrier**
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
Timing Namespace Reference
Detailed Description

The Nebula3 Timing Subsystem

The Nebula3 Timing subsystem offers classes and services for measuring elapsed time, and working with the calendar time (days, weeks, month, years).

Timing Subsystem Basics

- **Timing::Time** represents a double-precision floating point number and contains a time value in seconds.
- **Timing::Tick** represent an integer number contains a time value in milli-seconds (1/1000sec).
- To convert between the 2 data types, use the global **Timing::TicksToSeconds()** and **Timing::SecondsToTicks()** functions.
- To let a thread sleep for a specific amount of time, use the **Timing::Sleep()** method.
- Use **Timing::Sleep(0.0)** to give up the current thread time slice.
- Use the **Timing::Timer** class for measuring elapsed time. **Timing::Timer** is also "fast enough" to be used as a profiler around a block of code.
- Use the **Timing::CalendarTime** class if you need to query the current wall-clock time and date.
- The **Timing::CalendarTime** class also offers method to convert between calendar time and **IO::FileTime**.
Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>CalendarTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>Timer</td>
</tr>
</tbody>
</table>
## Typedefs

<table>
<thead>
<tr>
<th>Typedef</th>
<th>Datatype Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>typedef double Time</code></td>
<td><em>the time datatype</em></td>
</tr>
<tr>
<td><code>typedef int Tick</code></td>
<td><em>the tick datatype (one tick == 1 millisecond)</em></td>
</tr>
</tbody>
</table>
Functions

\begin{tabular}{ll}
\textbf{Time} & \textbf{TicksToSeconds} (\textbf{Tick} ticks) \\
\textbf{Tick} & \textbf{SecondsToTicks} (\textbf{Time} t) \\
\textbf{void} & \textbf{Sleep} (\textbf{Time} t)
\end{tabular}
Function Documentation

\texttt{Time}\nTiming::TicksToSeconds(Tick \textit{ticks}) \texttt{[inline]}

Convert ticks to seconds.

\texttt{Tick}\nTiming::SecondsToTicks(Time \textit{t}) \texttt{[inline]}

Convert seconds to ticks

\texttt{void}\nTiming::Sleep(Time \textit{t}) \texttt{[inline]}

Put current thread to sleep for specified amount of seconds.

The Nebula Device 3 documentation generated by \texttt{doxygen} at Fri Mar 26 15:21:50 2010
Util Namespace Reference
Detailed Description

TypePunning

Function to implement type-punning, explanation here: http://mail-index.netbsd.org/tech-kern/2003/08/11/0001.html
http://gcc.gnu.org/onlinedocs/gcc-4.1.1/gcc/Optimize-Options.html#Optimize-Options

(C) 2009 Radon Labs GmbH
# Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td><strong>Array</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>BitField</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>Blob</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>CommandLineArgs</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>Crc</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>Delegate</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>Dictionary</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>FixedArray</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>FixedTable</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>FourCC</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>GlobalStringAtomTable</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>HashTable</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>KeyValuePair</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>List</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>LocalStringAtomTable</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>Queue</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>RandomNumberTable</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>RingBuffer</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>RunLengthCodec</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>SimpleTree</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>SparseTable</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>Stack</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>String</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>StringAtom</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>StringAtomTableBase</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>StringBuffer</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>Variant</strong></td>
</tr>
<tr>
<td>class</td>
<td><strong>Guid</strong></td>
</tr>
</tbody>
</table>
Visibility Namespace Reference
Detailed Description

This file was generated with Nebula3's idlc compiler tool. DO NOT EDIT
Data Structures

- class `ObserverContext`
- class `VisibilityChecker`
- class `VisibilityContainer`
- class `VisibilityContext`
- class `VisibilityQuery`
- class `VisibilityBoxSystem`
- class `VisibilityClusterSystem`
- class `VisibilityQuadtree`
- class `VisibilitySystemBase`
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
Win32 Namespace Reference
Detailed Description

[TODO: Describe Win32 subsystem]
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysFunc</td>
<td></td>
</tr>
<tr>
<td>Win32MiniDump</td>
<td></td>
</tr>
<tr>
<td>Win32ConsoleHandler</td>
<td></td>
</tr>
<tr>
<td>Win32Cpu</td>
<td></td>
</tr>
<tr>
<td>Win32SystemInfo</td>
<td></td>
</tr>
<tr>
<td>Win32Guid</td>
<td></td>
</tr>
<tr>
<td>Win32StringConverter</td>
<td></td>
</tr>
<tr>
<td>Win32SkinnedMeshRenderer</td>
<td></td>
</tr>
<tr>
<td>Win32DisplayDevice</td>
<td></td>
</tr>
<tr>
<td>Win32InputDisplayEventHandler</td>
<td></td>
</tr>
<tr>
<td>Win32InputServer</td>
<td></td>
</tr>
<tr>
<td>Win32Mouse</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](https://www.doxygen.org/) at Fri Mar 26 15:21:50 2010
Win360 Namespace Reference
Detailed Description

(C) 2009 Radon Labs GmbH
### Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Win360FileTime</strong></td>
</tr>
<tr>
<td><strong>Win360FSWrapper</strong></td>
</tr>
<tr>
<td><strong>Win360Heap</strong></td>
</tr>
<tr>
<td><strong>Win360MemoryPool</strong></td>
</tr>
<tr>
<td><strong>Win360IpAddress</strong></td>
</tr>
<tr>
<td><strong>Win360Socket</strong></td>
</tr>
<tr>
<td><strong>Win360CriticalSection</strong></td>
</tr>
<tr>
<td><strong>Win360Event</strong></td>
</tr>
<tr>
<td><strong>Win360Interlocked</strong></td>
</tr>
<tr>
<td><strong>Win360Thread</strong></td>
</tr>
<tr>
<td><strong>Win360ThreadBarrier</strong></td>
</tr>
<tr>
<td><strong>Win360CalendarTime</strong></td>
</tr>
<tr>
<td><strong>Win360Timer</strong></td>
</tr>
<tr>
<td><strong>D3D9IndexBuffer</strong></td>
</tr>
<tr>
<td><strong>D3D9MemoryIndexBufferLoader</strong></td>
</tr>
<tr>
<td><strong>D3D9MemoryVertexBufferLoader</strong></td>
</tr>
<tr>
<td><strong>D3D9ShapeRenderer</strong></td>
</tr>
<tr>
<td><strong>D3D9StreamMeshLoader</strong></td>
</tr>
<tr>
<td><strong>D3D9StreamTextureSaver</strong></td>
</tr>
<tr>
<td><strong>D3D9TransformDevice</strong></td>
</tr>
<tr>
<td><strong>D3D9Types</strong></td>
</tr>
<tr>
<td><strong>D3D9VertexBuffer</strong></td>
</tr>
<tr>
<td><strong>D3D9VertexLayout</strong></td>
</tr>
<tr>
<td><strong>Win360Barrier</strong></td>
</tr>
<tr>
<td><strong>Win360ThreadId</strong></td>
</tr>
</tbody>
</table>
Here is a list of all documented namespace members with links to the namespaces they belong to:

- __Dispatcher() : **Messaging**
- __Handler() : **Messaging**
- __HeapAlignPointerAndWritePadding16() : **Memory**
- __HeapAlloc16() : **Memory**
- a -
  - Alloc() : Memory
  - AnimJobUtilMix() : CoreAnimation
  - AnimJobUtilSampleLinear() : CoreAnimation
  - AnimJobUtilSampleStep() : CoreAnimation

- c -
  - Clear() : Memory
  - Copy() : Memory
  - CopyToGraphicsMemory() : Memory

- d -
  - DumpTotalMemoryStatus() : Memory
  - DuplicateCString() : Memory

- e -
  - Exit() : Application

- f -
  - Fill() : Memory
  - Free() : Memory

- g -
  - GetTotalMemoryStatus() : Memory

- h -
- **Heaps** : Memory

- **i** -
  - IsOverlapping() : Memory

- **n** -
  - n_angulardistance() : Math
  - n_clamp() : Math
  - n_fchop() : Math
  - n_fequal() : Math
  - n_fgreater() : Math
  - n_fless() : Math
  - n_frnd() : Math
  - n_IClamp() : Math
  - n_isdenormal() : Math
  - n_lerp() : Math
  - n_log() : Math
  - n_log2() : Math
  - n_modangle() : Math
  - n_nearequal() : Math
  - n_rand() : Math
  - n_saturate() : Math
  - n_undenormalize() : Math

- **r** -
  - Realloc() : Memory

- **s** -
  - SecondsToTicks() : Timing
  - SetupHeaps() : Memory
  - Sleep() : Timing

- **t** -
- Tick : **Timing**
- TicksToSeconds() : **Timing**
- Time : **Timing**

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:51 2010
## The Nebula Device 3 File List

Here is a list of all documented files with brief descriptions:

<table>
<thead>
<tr>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>c:/nebula3/code/application/appgame/appconfig.h</td>
</tr>
<tr>
<td>c:/nebula3/code/application/basegamefeature/basegameattr/basegameattributes.h</td>
</tr>
<tr>
<td>c:/nebula3/code/application/graphicsfeature/graphicsfeatureproperties.h</td>
</tr>
<tr>
<td>c:/nebula3/code/application/physicsfeature/physicsattr/physicsattributes.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/foundation.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/core/config.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/core/debug.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/core/rttimacros.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/core/osx/osxsingleton.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/core/osx/precompiled.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/core/win32/precompiled.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/core/win32/win32singleton.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/io/util/bxmlfilestructs.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/jobs/stdjob.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/math/matrix44.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/math/scalar.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/memory/osx/osxmem.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/memory/osx/osxmemoryconfig.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/memory/win32/win32memory.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/memory/win360/win360memory.h</td>
</tr>
<tr>
<td>c:/nebula3/code/foundation/timing/time.h</td>
</tr>
<tr>
<td>c:/nebula3/code/render/stdneb.h</td>
</tr>
<tr>
<td>c:/nebula3/code/render/coreanimation/naxfileformatstructs.h</td>
</tr>
<tr>
<td>c:/nebula3/code/render/coreanimation/jobs/animjobutil.h</td>
</tr>
<tr>
<td>c:/nebula3/code/render/coregraphics/config.h</td>
</tr>
<tr>
<td>c:/nebula3/code/render/coregraphics/shadersemantics.h</td>
</tr>
<tr>
<td>c:/nebula3/code/render/graphics/stagebuilders.h</td>
</tr>
<tr>
<td>c:/nebula3/code/render/input/char.h</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
Detailed Description

Configuration header.

(C) 2007 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
Detailed Description

This is the central attribute registry for this feature. For more information on attributes, see Attr::Attribute.

(C) 2007 Radon Labs GmbH

#include "attr/attnid.h" #include "attr/attributedefinition.h"
Namespaces

namespace Attr
c:/nebula3/code/application/graphicsfeature
File Reference
Detailed Description

Top-level header file of GraphicsFeature.

(C) 2008 Radon Labs GmbH

#include "graphicsfeature/properties/cameraproperty.h"
#include "graphicsfeature/properties/chasecameraproperty.h"
#include "graphicsfeature/properties/mayacameraproperty.h"
#include "graphicsfeature/properties/environmentgraphicsproperty.h"
#include "graphicsfeature/properties/graphicsproperty.h"
#include "graphicsfeature/properties/inputproperty.h"
#include "graphicsfeature/properties/lightproperty.h"
#include "graphicsfeature/properties/actorgraphicsproperty.h"
#include "graphicsfeature/properties/ambiencebubbleproperty.h"
#include "graphicsfeature/properties/animationcontrolproperty.h"
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- File List
- Globals
Detailed Description

This is the central attribute registry for this feature. For more information on attributes, see Attr::Attribute.

(C) 2007 Radon Labs GmbH

#include "attr/attrid.h" #include "attr/attributedefinition.h"
Namespaces

namespace Attr

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:39 2010
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- File List
- Globals
Detailed Description

This is the central attribute registry for this feature. For more information on attributes, see Attr::Attribute.

(C) 2007 Radon Labs GmbH

#include "attr/attrid.h" #include "physicsattr/physicsattributes.h"
Namespaces

namespace Attr
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- File List
- Globals
c:/nebula3/code/foundation/foundation.h
File Reference
Detailed Description

Add all class headers for classes which need dynamic object creation by class name or class fourcc here.

(C) 2007 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
Detailed Description

Precompiled header. Put platform-specific headers which rarely change in here (e.g. windows.h).

(C) 2007 Radon Labs GmbH

#include "core/config.h"
c:/nebula3/code/foundation/core/config.h
File Reference
Detailed Description

Nebula3 compiler specific defines and configuration.

(C) 2006 Radon Labs GmbH
Variables

static const int JobMaxSliceSize = 0x3FF
Variable Documentation

const int JobMaxSliceSize = 0x3FFF [static]

max size of a data slice is 16 kByte - 1 byte this needs to be in a header, which is accessable from SPU too, thats why its here
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
File Reference
Detailed Description

Nebula debug macros.

n_assert() - the vanilla assert() Macro n_verify() - like assert() except that the statement in parens is simply evaluated, unchecked, if ___NEBULA_NO_ASSERT is set n_assert2() - an assert() plus a message from the programmer

#include "core/config.h"
## Functions

<table>
<thead>
<tr>
<th>void</th>
<th>void</th>
<th>void</th>
<th>void</th>
<th>void</th>
<th>void</th>
<th>n_sleep (double)</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>n_barf (const char *, const char *, int)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>n_barf2 (const char *, const char *, const char *, int)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Function Documentation

```c
void n_sleep ( double sec )
```

Put process to sleep.

- 21-Dec-98 floh created

```c
void n_barf ( const char * exp,
              const char * file,
              int line )
```

This function is called by n_assert() when the assertion fails.

```c
void n_barf2 ( const char * exp,
              const char * msg,
              const char * file,
              int line )
```

This function is called by n_assert2() when the assertion fails.
Detailed Description

This defines the macros for Nebula3's RTTI mechanism (__DeclareClass, __ImplementClass, etc...).

(C) 2008 Radon Labs GmbH
## Defines

<table>
<thead>
<tr>
<th>Define</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__DeclareClass(type)</td>
<td></td>
</tr>
<tr>
<td>__RegisterClass(type)</td>
<td>static const bool type##_registered = type::RegisterWithFactory();</td>
</tr>
<tr>
<td>__ImplementClass(type, fourcc, baseType)</td>
<td></td>
</tr>
<tr>
<td>__ImplementRootClass(type, fourcc)</td>
<td></td>
</tr>
</tbody>
</table>
Define Documentation

#define __DeclareClass (type )

Value:

```cpp
public: \
    void* operator new(size_t size) \n    { \n        return RTTI.AllocInstanceMemory(); \n    }; \n    void operator delete(void* p) \n    { \n        RTTI.FreeInstanceMemory(p); \n    }; \n    static Core::Rtti RTTI; \n    static Core::RefCounted* FactoryCreator(); \n    static type* Create(); \n    static bool RegisterWithFactory(); \n    virtual Core::Rtti* GetRtti() const; 
private:
```

Declaration macro. Put this into the class declaration.

#define __RegisterClass (type )

Value:

```cpp
static const bool type##_registered = type::RegisterWithFactory();
```

Register a class with the factory. This is only necessary for classes which can create objects by name or fourcc.

#define __ImplementClass (type, fourcc, baseType )

Value:

```cpp
Core::Rtti type::RTTI(#type, fourcc, type::FactoryCreator, &baseTyp
Core::Rtti* type::GetRtti() const { return &this->RTTI; } \nCore::RefCounted* type::FactoryCreator() { return type::Create( 
type* type::Create() \n```
Implementation macro for default memory pool sizes. Put this into the source file.

```
#define __ImplementRootClass(type, fourcc)

Value:
```

```
Core::Rtti type::RTTI(#type, fourcc, type::FactoryCreator, 0, sizeof
Core::Rtti* type::GetRtti() const { return &this->RTTI; } 
Core::RefCounted* type::FactoryCreator() { return type::Create(); }
{ 
    return n_new(type); 
}
bool type::RegisterWithFactory() 
{ 
    if (!Core::Factory::Instance()->ClassExists(#type)) 
    { 
        Core::Factory::Instance()->Register(&type::RTTI, #type,
    } 
    return true; 
}
```

Type implementation of topmost type in inheritance hierarchy (source file).
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
Detailed Description

Provides helper macros to implement singleton objects:

- __DeclareSingleton put this into class declaration
- __ImplementSingleton put this into the implementation file
- __ConstructSingleton put this into the constructor
- __DestructSingleton put this into the destructor

Get a pointer to a singleton object using the static Instance() method:

Core::Server* coreServer = Core::Server::Instance();

#include "core/types.h" #include "threading/osx/osxthreadlocalptr.h"

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:39 2010
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- File List
- Globals
Detailed Description

Contains precompiled headers on the OSX platform.

(C) 2010 Radon Labs GmbH

#include <stdlib.h> #include <stdio.h>
#include <math.h>
#include <stdarg.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
#include <pthread.h>
#include <malloc/malloc.h>
#include <uuid/uuid.h>
#include <algorithm>

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:39 2010
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- File List
- Globals
c:/nebula3/code/foundation/core/win32/pr
File Reference
Detailed Description

Contains precompiled headers on the Win32 platform.

(C) 2007 Radon Labs GmbH

#include <windows.h> #include <winbase.h>
#include <process.h>
#include <shfolder.h>
#include <tchar.h>
#include <strsafe.h>
#include <wininet.h>
#include <winsock2.h>
#include <rpc.h>
#include <dbghelp.h>
#include <intrin.h>
#include <xnamath.h>
#include <math.h>
#include <stdlib.h>
#include <stdarg.h>
#include <algorithm>
#include <d3d9.h>
#include <d3dx9.h>
#include <dxerr.h>
#include <xinput.h>
#include <xact3.h>
#include <xact3d3.h>
#include <dinput.h>
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
c:/nebula3/code/foundation/core/win32/wi
File Reference
Detailed Description

Provides helper macros to implement singleton objects:

- __DeclareSingleton put this into class declaration
- __ImplementSingleton put this into the implemention file
- __ConstructSingleton put this into the constructor
- __DestructSingleton put this into the destructor

Get a pointer to a singleton object using the static Instance() method:

Core::Server* coreServer = Core::Server::Instance();

(C) 2007 Radon Labs GmbH

#include "core/types.h"
File Reference
Detailed Description

Structures used by the BXML file format.

(C) 2009 Radon Labs GmbH

#include "core/types.h"
Namespaces

namespace IO
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
c:/nebula3/code/foundation/jobs/stdjob.h
File Reference
Detailed Description

Standard header for job functions.

(C) 2009 Radon Labs GmbH

#include "core/config.h" #include "core/win32/precompiled.h"
#include "jobs/jobfunccontext.h"
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- File List
- Globals
Detailed Description

Frontend header for matrix classes.

(C) 2006 Radon Labs GmbH

#include "math/xnamath/xna_matrix44.h"
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
Detailed Description

Nebula's scalar datatype.

NOTE: do not add CRT math function calls to this call, but instead into the platform specific headers (for instance, on the Wii the sinc() functions are called and must be placed into a .cc file, not into the header.

(C) 2007 Radon Labs GmbH

#include "math/xnamath/xna_scalar.h"
Namespaces

```csharp
namespace Math
```
## Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__forceinline bool</td>
<td>Math::n_fequal</td>
<td>(scalar f0, scalar f1, scalar tol)</td>
</tr>
<tr>
<td>__forceinline int</td>
<td>Math::n_iclamp</td>
<td>(int val, int minVal, int maxVal)</td>
</tr>
<tr>
<td>__forceinline bool</td>
<td>Math::n_fless</td>
<td>(scalar f0, scalar f1, scalar tol)</td>
</tr>
<tr>
<td>__forceinline bool</td>
<td>Math::n_fgreater</td>
<td>(scalar f0, scalar f1, scalar tol)</td>
</tr>
<tr>
<td>__forceinline scalar</td>
<td>Math::n Clamp</td>
<td>(scalar val, scalar lower, scalar upper)</td>
</tr>
<tr>
<td>__forceinline scalar</td>
<td>Math::n_saturate</td>
<td>(scalar val)</td>
</tr>
<tr>
<td>__forceinline double</td>
<td>Math::n_saturate</td>
<td>(double val)</td>
</tr>
<tr>
<td>__forceinline scalar</td>
<td>Math::n_lerp</td>
<td>(scalar x, scalar y, scalar l)</td>
</tr>
<tr>
<td>__forceinline double</td>
<td>Math::n_lerp</td>
<td>(double x, double y, double l)</td>
</tr>
<tr>
<td>__forceinline scalar</td>
<td>Math::n_angulardistance</td>
<td>(scalar from, scalar to)</td>
</tr>
<tr>
<td>__forceinline bool</td>
<td>Math::n_isdenormal</td>
<td>(scalar s)</td>
</tr>
<tr>
<td>__forceinline float</td>
<td>Math::n_undenormalize</td>
<td>(scalar s)</td>
</tr>
<tr>
<td>__forceinline bool</td>
<td>Math::n_nearequal</td>
<td>(scalar a, scalar b, scalar epsilon)</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Fri Mar 26 15:21:39 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
c:/nebula3/code/foundation/memory/osx/
File Reference
Detailed Description

Lowlevel memory functions for the OSX platform.

(C) 2010 Radon Labs GmbH

#include "core/config.h" #include "core/debug.h"
#include "memory/osx/osxmemoryconfig.h"
Namespaces

namespace Memory
Data Structures

```c
struct Memory::TotalMemoryStatus
```
Defines

#define __MEMORY_CHECKPOINT(s)
### Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory::Alloc</td>
<td>Allocate a chunk of memory</td>
</tr>
<tr>
<td>Memory::Realloc</td>
<td>Re-allocate a chunk of memory</td>
</tr>
<tr>
<td>Memory::Free</td>
<td>Free a chunk of memory</td>
</tr>
<tr>
<td>Memory::DuplicateCString</td>
<td>Duplicate a C-string (obsolete)</td>
</tr>
<tr>
<td>Memory::IsOverlapping</td>
<td>Check if 2 memory regions are overlapping</td>
</tr>
<tr>
<td>Memory::Copy</td>
<td>Copy a chunk of memory</td>
</tr>
<tr>
<td>Memory::Clear</td>
<td>Overwrite a chunk of memory with zero</td>
</tr>
<tr>
<td>Memory::Fill</td>
<td>Fill memory with a specific byte</td>
</tr>
<tr>
<td>Memory::GetTotalMemoryStatus</td>
<td>Get total memory status</td>
</tr>
<tr>
<td>operator new</td>
<td></td>
</tr>
</tbody>
</table>

### TotalMemoryStatus

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory::GetTotalMemoryStatus</td>
<td></td>
</tr>
<tr>
<td>operator new</td>
<td></td>
</tr>
</tbody>
</table>

*HeapType heapType, size_t size, size_t alignment=16* 
*HeapType heapType, size_t size* 
*HeapType heapType, void *ptr, size_t size* 
*HeapType heapType, void *ptr* 
*const char *from* 
*const unsigned char *srcPtr, size_t srcSize, const unsigned char *dstPtr, size_t dstSize* 
*const void *from, void *to, size_t numBytes* 
*void *ptr, size_t numBytes* 
*void *ptr, size_t numBytes, unsigned char value*
Define Documentation

#define __MEMORY_CHECKPOINT(s)

Debug and memory validation functions.
Function Documentation

void* operator ( size_t size )
new

Replacement global new/delete operators.
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
Detailed Description

Central config file for memory setup on the OSX platform.

(C) 2010 Radon Labs GmbH

#include "core/config.h"
Namespaces

namespace Memory
Enumerations

```cpp
enum Memory::HeapType
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void Memory::SetupHeaps ()</code></td>
<td></td>
</tr>
<tr>
<td><code>const char * Memory::GetHeapTypeName (HeapType heapType)</code></td>
<td></td>
</tr>
</tbody>
</table>
Variables

malloc_zone_t * Memory::Heaps [NumHeapTypes]
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
Detailed Description

Win32 specific low level memory functions.

(C) 2006 Radon Labs GmbH

#include "core/config.h" #include "core/debug.h"
Namespaces

namespace Memory
### Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>Memory::Copy</code></td>
<td>(const void *from, void *to, size_t numBytes) <code>copy a chunk of memory</code></td>
</tr>
<tr>
<td>__forceinline void</td>
<td><code>Memory::CopyToGraphicsMemory</code></td>
<td>(const void *from, void *to, size_t numBytes) <code>copy a chunk of memory</code></td>
</tr>
<tr>
<td>void</td>
<td><code>Memory::Clear</code></td>
<td>(void *ptr, size_t numBytes) <code>overwrite a chunk of memory with zero</code></td>
</tr>
<tr>
<td>void</td>
<td><code>Memory::Fill</code></td>
<td>(void *ptr, size_t numBytes, unsigned char value) <code>fill memory with a specific byte</code></td>
</tr>
</tbody>
</table>
Detailed Description

Memory subsystem features which are identical on Win32 and Xbox360.

(C) 2008 Radon Labs GmbH

#include "core/config.h" #include "core/debug.h"
#include "threading/interlocked.h"
#include "memory/win360/win360memoryconfig.h"
#include "memory/win32/win32memory.h"
Namespaces

namespace Memory
Data Structures

```c
struct Memory::TotalMemoryStatus
```
Defines

#define __MEMORY_CHECKPOINT(s)
### Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void * Memory::Alloc (HeapType heapType, size_t size)</code></td>
<td>allocate a chunk of memory</td>
</tr>
<tr>
<td><code>void * Memory::Realloc (HeapType heapType, void *ptr, size_t size)</code></td>
<td>re-allocate a chunk of memory</td>
</tr>
<tr>
<td><code>void Memory::Free (HeapType heapType, void *ptr)</code></td>
<td>free a chunk of memory</td>
</tr>
<tr>
<td><code>char * Memory::DuplicateCString (const char *from)</code></td>
<td>duplicate a C-string (obsolete)</td>
</tr>
<tr>
<td><code>bool Memory::IsOverlapping (const unsigned char *srcPtr, size_t srcSize, const unsigned char *dstPtr, size_t dstSize)</code></td>
<td>check if 2 memory regions are overlapping</td>
</tr>
<tr>
<td><code>TotalMemoryStatus Memory::GetTotalMemoryStatus ()</code></td>
<td></td>
</tr>
<tr>
<td><code>void Memory::DumpTotalMemoryStatus ()</code></td>
<td></td>
</tr>
<tr>
<td><code>__forceinline void * operator new (size_t size)</code></td>
<td></td>
</tr>
</tbody>
</table>
Define Documentation

#define __MEMORY_CHECKPOINT(s)

Debug function which validates the process heap. This will NOT check local heaps (call Heap::ValidateAllHeaps() for this). Stops the program if something is wrong.
Function Documentation

__forceinline
void* operator new ( size_t size )
new

Replacement global new/delete operators.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- File List
- Globals
Detailed Description

Central config file for memory setup on the Win32 and Xbox360 platform.

(C) 2008 Radon Labs GmbH

#include "core/config.h" #include "core/debug.h"
Namespaces

namespace Memory
Enumerations

```cpp
enum Memory::HeapType
```
### Functions

<table>
<thead>
<tr>
<th>void</th>
<th>Memory::SetupHeaps ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>const char *</td>
<td>Memory::GetHeapTypeName (HeapType heapType)</td>
</tr>
<tr>
<td>__forceinline unsigned char *</td>
<td>Memory::__HeapAlignPointerAndWritePadding16 (unsigned char *ptr)</td>
</tr>
<tr>
<td>__forceinline unsigned char *</td>
<td>Memory::__HeapUnalignPointer16 (unsigned char *ptr)</td>
</tr>
<tr>
<td>__forceinline LPVOID</td>
<td>Memory::__HeapAlloc16 (HANDLE hHeap, DWORD dwFlags, SIZE_T dwBytes)</td>
</tr>
<tr>
<td>__forceinline LPVOID</td>
<td>Memory::__HeapReAlloc16 (HANDLE hHeap, DWORD dwFlags, LPVOID lpMem, SIZE_T dwBytes)</td>
</tr>
<tr>
<td>__forceinline BOOL</td>
<td>Memory::__HeapFree16 (HANDLE hHeap, DWORD dwFlags, LPVOID lpMem)</td>
</tr>
<tr>
<td>__forceinline SIZE_T</td>
<td>Memory::__HeapSize16 (HANDLE hHeap, DWORD dwFlags, LPCVOID lpMem)</td>
</tr>
</tbody>
</table>
Variables

HANDLE volatile Memory::Heaps [NumHeapTypes]
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- File List
- Globals
Detailed Description

Typedefs for the **Timing** subsystem

(C) 2006 Radon Labs GmbH

#include "core/types.h"
Namespaces

namespace Timing
### Typedefs

<table>
<thead>
<tr>
<th>typedef double Timing::Time</th>
<th>the time datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>typedef int Timing::Tick</td>
<td>the tick datatype (one tick == 1 millisecond)</td>
</tr>
</tbody>
</table>
Functions

```
Time  Timing::TicksToSeconds (Tick ticks)
Tick  Timing::SecondsToTicks (Time t)
void  Timing::Sleep (Time t)
```
Detailed Description

Precompiled header. Put platform-specific headers which rarely change in here (e.g. windows.h).

(C) 2007 Radon Labs GmbH

#include "core/config.h" #include "core/win32/precompiled.h"
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
Detailed Description

NAX file format structures.

(C) 2009 Radon Labs GmbH

#include "core/types.h"
Namespaces

namespace CoreAnimation
Data Structures

struct CoreAnimation::Nax3Header
struct CoreAnimation::Nax2Header

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:39 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
Detailed Description

Utility functions for CoreAnimation subsystem jobs.

(C) 2009 Radon Labs GmbH

#include "math/float4.h" #include "math/quaternion.h"
#include "coreanimation/animcurve.h"
Namespaces

namespace CoreAnimation
## Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CoreAnimation::AnimJobUtilSampleStep</code></td>
<td>(const AnimCurve *curves, int numCurves, const float4 &amp;velocityScale, const float4 *src0SamplePtr, float4 *outSamplePtr, uchar *outSampleCounts)</td>
</tr>
<tr>
<td><code>CoreAnimation::AnimJobUtilSampleLinear</code></td>
<td>(const AnimCurve *curves, int numCurves, float sampleWeight, const float4 &amp;velocityScale, const float4 *src0SamplePtr, const float4 *src1SamplePtr, float4 *outSamplePtr, uchar *outSampleCounts)</td>
</tr>
<tr>
<td><code>CoreAnimation::AnimJobUtilMix</code></td>
<td>(const AnimCurve *curves, int numCurves, float mixWeight, const float4 *src0SamplePtr, const float4 *src1SamplePtr, const uchar *src0SampleCounts, const uchar *src1SampleCounts, float4 *outSamplePtr, uchar *outSampleCounts)</td>
</tr>
</tbody>
</table>
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- File List
- Globals
Detailed Description

Compile time configuration options for the CoreGraphics subsystem.

(C) 2007 Radon Labs GmbH

#include "core/types.h"
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
Detailed Description

Standard shader variable semantic names.

(C) 2009 Radon Labs GmbH

#include "core/types.h"
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- File List
- Globals
Detailed Description

Frontend typedefs to InternalGraphics::StageBuilder classes.

(C) 2008 Radon Labs GmbH

#include "internalgraphics/stagebuilder.h" #include "internalgraphics/simplestagebuilder.h" #include "internalgraphics/quadtreestagebuilder.h"
Namespaces

namespace Graphics

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:39 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals
c:/nebula3/code/render/input/char.h File Reference
Detailed Description

A translated character code.

(C) 2006 Radon Labs GmbH

#include "core/types.h"
Namespaces

namespace Input
- __HeapAlignPointerAndWritePadding16() : win360memoryconfig.h
- __HeapAlloc16() : win360memoryconfig.h
- __HeapFree16() : win360memoryconfig.h
- __HeapReAlloc16() : win360memoryconfig.h
- __HeapSize16() : win360memoryconfig.h
- __HeapUnalignPointer16() : win360memoryconfig.h
- a -
  - AnimJobUtilMix() : animjobutil.h
  - AnimJobUtilSampleLinear() : animjobutil.h
  - AnimJobUtilSampleStep() : animjobutil.h

- c -
  - CopyToGraphicsMemory() : win32memory.h

- h -
  - Heaps : osxmemoryconfig.h, win360memoryconfig.h

- j -
  - JobMaxSliceSize : config.h

- n -
  - n_angulardistance() : scalar.h
  - n_clamp() : scalar.h
  - n_fequal() : scalar.h
  - n_fgreater() : scalar.h
  - n_fless() : scalar.h
  - n_iclamp() : scalar.h
  - n_isdenormal() : scalar.h
  - n_lerp() : scalar.h
  - n_nearequal() : scalar.h
  - n_saturate() : scalar.h
  - n_undenormalize() : scalar.h

- o -
  - operator new() : osxmemory.h

- s -
- t -

- SecondsToTicks() : `time.h`
- Sleep() : `time.h`
- Tick : `time.h`
- TicksToSeconds() : `time.h`
- Time : `time.h`
The Nebula Device 3 Related Pages

Here is a list of all related documentation pages:

- Todo List
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages
Todo List

Global `Base::ShaderVariableInstanceBase::Apply()`:
  hmm, the dynamic type switch is sort of lame...

Class `Direct3D9::D3D9Shader`
  lost/reset device handling

Class `Direct3D9::D3D9ShaderInstance`
  lost/reset device handling

Global `InternalGraphics::InternalView::ResolveVisibleLights()`:
  currently this methods needs to go over all visible graphics entities to find the lights...

Class `InternalGraphics::VisibilityCell`
  need to handle extra shadow bounding box
    : statistics and profiling
    : need to add visibility depending on LOD
    : multithreaded visibility link update?
    : add dirty handling to visibility links (e.g. don't need to update links between static lights and static objects)

Class `IO::BinaryReader`
  convert endianess!

Class `IO::BinaryWriter`
  convert endianess!

Global `Lighting::SM30LightServer::ApplyModelEntityLights(const Ptr<InternalGraphics::InternalModelEntity>& modelEntity)`:
  set light properties only once per-frame and only set a bool array with active per-model-entity-lights here!

Class `Math::bbox`
: UNTESTED!

Global **Math::line::intersect** (const line &l, point &pa, point &pb) const
: Untested! Replace with simpler code.

Class **OSX::OSXCriticalSection**
: Add debugging asserts? If yes wrap with new __NEBULA3-declare

Class **OSX::OSXThreadLocalPtr< TYPE >**
: Performance? Would it actually be better to allocate one pointer lookup-table and associate this with a single thread-local key as like on the Wii? At the moment every **Ptr** has its own key.

Class **OSX::OSXThreadLocalPtr< TYPE >**
: If this object will be used for anything else then singleton, it might make sense to change the interface to look like a normal C-pointer.

Class **Resources::ResourceDictionary**
: ResourceDictionary should be the base to switch to numerical resource id's later on. During asset export, an offline-dictionary is kept which associated actual filenames with numerical id's, and the asset export tools write those numerical ids in place where a string resource name would be used. For now it's just a lookup-table for the resource size, which is only really useful for console-platforms (since on those platforms the application has more control over the resource loading process as compared to PC-APIs).

Class **Threading::Event**
describe Event class

Class **Threading::Thread**
describe Thread class

Global **Util::Array::Difference** (const Array< TYPE > &rhs)
this method is broken, check test case to see why!
Global `Util::FixedArray::BinarySearchIndex` (const TYPE &val)
const
    hmm, this is copy-pasted from Array...

Class `Win360::D3D9StreamMeshLoader`
    : document file formats

Class `Win360::Win360Timer`
    solve multiprocessor issues of QueryPerformanceCounter()
    (different processors may return different PerformanceFrequency
    values, thus, threads should be prevented from switching
    between processors with thread affinities).
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields
# The Nebula Device 3 Data Structure Index

<p>| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
| <strong>AbstractLightEntity (Graphics)</strong> | <strong>Action (Actions)</strong> | <strong>ActionList (Actions)</strong> | <strong>ActionReader (Script)</strong> | <strong>ActorGraphicsProperty (GraphicsFeature)</strong> | <strong>ActorPhysicsProperty (PhysicsFeature)</strong> | <strong>Adapter (CoreGraphics)</strong> | <strong>AdapterInfo (CoreGraphics)</strong> | <strong>And (Conditions)</strong> | <strong>AngularPFeedbackLoop</strong> | <strong>AnimationControlProperty (GraphicsFeature)</strong> | <strong>AnimatorInstance (Animator)</strong> | <strong>AnimLoopType (Animator)</strong> | <strong>AnimatorNode</strong> | <strong>DebugPacket (Net)</strong> | <strong>DebugPageHandler (Debug)</strong> | <strong>DebugServer (Debug)</strong> | <strong>DebugShapeRenderer (Debug)</strong> | <strong>DebugTextRenderer (Debug)</strong> | <strong>DebugTimer (Debug)</strong> | <strong>DefaultHttpRequestHandler (Http)</strong> | <strong>Delegate (Util)</strong> | <strong>DelegateTable (Messaging)</strong> | <strong>Dialog (Script)</strong> | <strong>DialogDesc (Script)</strong> | <strong>DialogManager (Script)</strong> | <strong>DialogTake (Script)</strong> | <strong>Dictionary (Util)</strong> | <strong>Dispatcher (Messaging)</strong> | <strong>IpAddress (Net)</strong> |</p>
<table>
<thead>
<tr>
<th>Class/Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnimEventServer</td>
<td>(Animation)</td>
</tr>
<tr>
<td>DisplaySettings</td>
<td>(Graphics)</td>
</tr>
<tr>
<td>JobPort</td>
<td>(Jobs)</td>
</tr>
<tr>
<td>AnimJob</td>
<td>(Animation)</td>
</tr>
<tr>
<td>DrawFullScreenQuad</td>
<td>(RenderUtil)</td>
</tr>
<tr>
<td>JobPortBase</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>AnimJobEnqueueMode</td>
<td>(Animation)</td>
</tr>
<tr>
<td>EmitterAttrs</td>
<td>(Particles)</td>
</tr>
<tr>
<td>JobServerBase</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>AnimKey</td>
<td>(Animation)</td>
</tr>
<tr>
<td>EmitterMesh</td>
<td>(Particles)</td>
</tr>
<tr>
<td>JobSystem</td>
<td>(Jobs)</td>
</tr>
<tr>
<td>AnimKeyArray</td>
<td>(Animation)</td>
</tr>
<tr>
<td>Entity</td>
<td>(Game)</td>
</tr>
<tr>
<td>JobUniformDesc</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>AnimKeyBuffer</td>
<td>(CoreAnimation)</td>
</tr>
<tr>
<td>EnvelopeCurve</td>
<td>(Particles)</td>
</tr>
<tr>
<td>KeyValuePair</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>AnimResource</td>
<td>(CoreAnimation)</td>
</tr>
<tr>
<td>EnvelopeSampleBuffer</td>
<td>(Particles)</td>
</tr>
<tr>
<td>LevelLoader</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>AnimSampleBuffer</td>
<td>(CoreAnimation)</td>
</tr>
<tr>
<td>EnvEntityManager</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>LightFlickerL</td>
<td>(GraphicsFeature)</td>
</tr>
<tr>
<td>AnimSampleMixInfo</td>
<td>(CoreAnimation)</td>
</tr>
<tr>
<td>EnvironmentCollideProperty</td>
<td>(PhysicsFeature)</td>
</tr>
<tr>
<td>LightingMode</td>
<td>(Frame)</td>
</tr>
<tr>
<td>AnimSequencer</td>
<td>(Animation)</td>
</tr>
<tr>
<td>EnvironmentLoader</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>LightPrePassServer</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>AnimUtil</td>
<td>(CoreAnimation)</td>
</tr>
<tr>
<td>EntityLoaderBase</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>LightServer</td>
<td>(Lighting)</td>
</tr>
<tr>
<td>AntiAliasQuality</td>
<td>(CoreGraphics)</td>
</tr>
<tr>
<td>EnvQueryManager</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>LightServerBase</td>
<td>(Lighting)</td>
</tr>
<tr>
<td>AppEntry</td>
<td>(System)</td>
</tr>
<tr>
<td>Event</td>
<td>(Threading)</td>
</tr>
<tr>
<td>Array</td>
<td>(Util)</td>
</tr>
<tr>
<td>ExcelXmlReader</td>
<td>(IO)</td>
</tr>
<tr>
<td>LightType</td>
<td>(Light)</td>
</tr>
<tr>
<td>Assign</td>
<td>(IO)</td>
</tr>
<tr>
<td>ExitHandler</td>
<td>(Core)</td>
</tr>
<tr>
<td>line</td>
<td>(Math)</td>
</tr>
<tr>
<td>AssignRegistry</td>
<td>(IO)</td>
</tr>
<tr>
<td>Extrapolator</td>
<td>(Math)</td>
</tr>
<tr>
<td>List</td>
<td>(Util)</td>
</tr>
<tr>
<td>AsyncPort</td>
<td>(Messaging)</td>
</tr>
<tr>
<td>Factory</td>
<td>(Core)</td>
</tr>
<tr>
<td>List::Iterator</td>
<td>(Util)</td>
</tr>
<tr>
<td>AttachmentManager</td>
<td>(GraphicsFeature)</td>
</tr>
<tr>
<td>FactoryManager</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>LoaderServer</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>AttachmentServer</td>
<td>(InternalGraphics)</td>
</tr>
<tr>
<td>FactoryManager::PropertyEntry</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>LoadingResource</td>
<td>(Resources)</td>
</tr>
<tr>
<td>Attributes</td>
<td>(Script)</td>
</tr>
<tr>
<td>FeatureUnit</td>
<td>(Game)</td>
</tr>
<tr>
<td>LogFileConsoleHandler</td>
<td></td>
</tr>
<tr>
<td>Archive</td>
<td>(IO)</td>
</tr>
<tr>
<td>ExcelXmlReader</td>
<td>(IO)</td>
</tr>
<tr>
<td>LocalStringAtomTable</td>
<td></td>
</tr>
<tr>
<td>ArchiveBase</td>
<td>(IO)</td>
</tr>
<tr>
<td>EnvQueryManager</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>LightType</td>
<td>(Light)</td>
</tr>
<tr>
<td>ArchiveFileSystem</td>
<td>(IO)</td>
</tr>
<tr>
<td>Event</td>
<td>(Threading)</td>
</tr>
<tr>
<td>LightType</td>
<td>(Light)</td>
</tr>
<tr>
<td>ArchiveFileSystemBase</td>
<td>(Base)</td>
</tr>
<tr>
<td>ExcelXmlReader</td>
<td>(IO)</td>
</tr>
<tr>
<td>LocalStringAtomTable</td>
<td></td>
</tr>
<tr>
<td>Array</td>
<td>(Util)</td>
</tr>
<tr>
<td>ExitHandler</td>
<td>(Core)</td>
</tr>
<tr>
<td>line</td>
<td>(Math)</td>
</tr>
<tr>
<td>Assign</td>
<td>(IO)</td>
</tr>
<tr>
<td>Extrapolator</td>
<td>(Math)</td>
</tr>
<tr>
<td>List</td>
<td>(Util)</td>
</tr>
<tr>
<td>AssignRegistry</td>
<td>(IO)</td>
</tr>
<tr>
<td>Factory</td>
<td>(Core)</td>
</tr>
<tr>
<td>List::Iterator</td>
<td>(Util)</td>
</tr>
<tr>
<td>AsyncPort</td>
<td>(Messaging)</td>
</tr>
<tr>
<td>FactoryManager</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>LoadingResource</td>
<td>(Resources)</td>
</tr>
<tr>
<td>AttachmentManager</td>
<td>(GraphicsFeature)</td>
</tr>
<tr>
<td>FactoryManager::PropertyEntry</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>LoadingResource</td>
<td>(Resources)</td>
</tr>
<tr>
<td>AttachmentServer</td>
<td>(InternalGraphics)</td>
</tr>
<tr>
<td>FactoryManager::PropertyEntry</td>
<td>(BaseGameFeature)</td>
</tr>
<tr>
<td>LoadingResource</td>
<td>(Resources)</td>
</tr>
<tr>
<td>Attributes</td>
<td>(Script)</td>
</tr>
<tr>
<td>FeatureUnit</td>
<td>(Game)</td>
</tr>
<tr>
<td>LogFileConsoleHandler</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>M</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>FileStream (IO)</strong></td>
<td><strong>ManagedAnimation (CoreAnimation)</strong></td>
</tr>
<tr>
<td><strong>Barrier (Threading)</strong></td>
<td><strong>FileTime (IO)</strong></td>
</tr>
<tr>
<td><strong>BaseGameFeatureUnit (Game)</strong></td>
<td><strong>FixedArray (Util)</strong></td>
</tr>
<tr>
<td><strong>BatchMessage (Messaging)</strong></td>
<td><strong>FixedTable (Util)</strong></td>
</tr>
<tr>
<td><strong>BatchType (CoreGraphics)</strong></td>
<td><strong>float2 (Math)</strong></td>
</tr>
<tr>
<td><strong>bbox (Math)</strong></td>
<td><strong>float4 (Math)</strong></td>
</tr>
<tr>
<td><strong>BinaryReader (IO)</strong></td>
<td><strong>FocusManager (BaseGameFeature)</strong></td>
</tr>
<tr>
<td><strong>BinaryWriter (IO)</strong></td>
<td><strong>FourCC (Util)</strong></td>
</tr>
<tr>
<td><strong>BitField (Util)</strong></td>
<td><strong>FrameBatch (Frame)</strong></td>
</tr>
<tr>
<td><strong>Blob (Util)</strong></td>
<td><strong>FramePass (Frame)</strong></td>
</tr>
<tr>
<td><strong>BlockingHandlerThread (Messaging)</strong></td>
<td><strong>FramePassBase (Frame)</strong></td>
</tr>
<tr>
<td><strong>BXmlLoaderUtil (IO)</strong></td>
<td><strong>FramePostEffect (Frame)</strong></td>
</tr>
<tr>
<td><strong>BXmlReader (IO)</strong></td>
<td><strong>FrameServer (Frame)</strong></td>
</tr>
<tr>
<td><strong>ByteOrder (System)</strong></td>
<td><strong>FrameShader (Frame)</strong></td>
</tr>
<tr>
<td><strong>CalendarTime (Timing)</strong></td>
<td><strong>FrameShaderLoader (Frame)</strong></td>
</tr>
<tr>
<td><strong>CalendarTimeBase (Base)</strong></td>
<td><strong>Memory (Memory)</strong></td>
</tr>
<tr>
<td><strong>CameraEntity (Graphics)</strong></td>
<td><strong>FrameSyncHandlerThread (FrameSync)</strong></td>
</tr>
<tr>
<td><strong>CameraProperty (GraphicsFeature)</strong></td>
<td><strong>FrameSyncSharedData (FrameSync)</strong></td>
</tr>
<tr>
<td><strong>CameraSettings (Shared)</strong></td>
<td><strong>FrameSyncTimer (FrameSync)</strong></td>
</tr>
<tr>
<td><strong>CategoryManager (BaseGameFeature)</strong></td>
<td><strong>frustum (Math)</strong></td>
</tr>
<tr>
<td><strong>CategoryManager::Category (BaseGameFeature)</strong></td>
<td><strong>FSMAction (Actions)</strong></td>
</tr>
<tr>
<td><strong>CategoryManager::Entry (BaseGameFeature)</strong></td>
<td><strong>FSMCondition (Conditions)</strong></td>
</tr>
<tr>
<td><strong>Character (Characters)</strong></td>
<td><strong>FSWrapper (IO)</strong></td>
</tr>
<tr>
<td><strong>CharacterAnimationController (Characters)</strong></td>
<td><strong>MemoryPool (Memory)</strong></td>
</tr>
<tr>
<td><strong>CharacterInstance (Characters)</strong></td>
<td><strong>MemoryStream</strong></td>
</tr>
<tr>
<td><strong>GameApplication (App)</strong></td>
<td><strong>MemoryVertexBufferLoader (CoreGraphics)</strong></td>
</tr>
<tr>
<td><strong>GameContentServer (IO)</strong></td>
<td><strong>MemoryVertexBufferLoader (Base)</strong></td>
</tr>
<tr>
<td><strong>GameContentServerBase (Base)</strong></td>
<td><strong>Mesh (CoreGraphics)</strong></td>
</tr>
<tr>
<td><strong>GameMesh (CoreGraphics)</strong></td>
<td><strong>MeshBase (Base)</strong></td>
</tr>
</tbody>
</table>
D3D9Texture (Direct3D9) InfoLog (Script) OSXThreadld (OSX ThreadId)
D3D9TextureStreamer (Resources) InfoLogFilter (Script) OSXThreadLocalPtr
D3D9TransformDevice (Win360) InputEvent (Input) Particle (ParticleRenderer)
D3D9Types (Win360) InputHandler (Input) ParticleRenderer (ParticleRenderer)
D3D9VertexBuffer (Win360) InputPriority (Input) ParticleRendererBase
D3D9VertexBufferLayout (Win360) InputServer (Input) ParticleRendererInfo (ParticleServer)
DebugCounter (Debug) InputServerBase (Base) ParticleServer (ParticleSystem)
DebugFloat (Debug) InputTimeSource (BaseGameFeature) ParticleServer (ParticleSystem)
DebugGraphicsHandler (Debug) Interface (IO) ParticleSystem (ParticleSystem)
DebugHandler (Debug) InterfaceBase (Interface) ParticleSystemInstance (ParticleSystem)
DebugInterface (Debug) InterfaceHandlerBase (Interface) ParticleSystemNode (ParticleSystem)
DebugMessage (Net) Interlocked (Threading) The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
__declspec() : Math::float4, Math::plane
- a -
- AutoEvadeProbeAboveGround : 
  PhysicsFeature::ActorPhysicsProperty
- AutoEvadeProbeRadius : 
  PhysicsFeature::ActorPhysicsProperty

- BEGINLABEL : Script::InfoLog

- DefaultStepTime : Particles::ParticleSystem

- ENDLABEL : Script::InfoLog

- HeaderSize : Net::DebugPacket

- InvalidCoreId : OSX::OSXCpu, Win32::Win32Cpu
  InvalidModelNodeType : Models::ModelNodeType

- managedResource : Resources::LoadingResource
  MaxNumModelNodeType : Models::ModelNodeType
  MaxNumRenderTarget : Base::MultipleRenderTargetBase
  MaxNumVertexStreams : Base::RenderDeviceBase
  MaxPayloadSize : Net::DebugPacket
  MaxSliceSize : Base::JobBase

- newSlot : Resources::PoolLoadingResource
- o -
  - oldSlot : Resources::PoolLoadingResource

- p -
  - PacketSize : Net::DebugPacket
  - poolMapper : Resources::TexturePoolMapperScheduler

- r -
  - resourceStreamingLevelOfDetail : Models::ModelNode

- s -
  - soundName : StateObjectFeature::StateInfo
  - StepTime : Particles::ParticleSystem

- t -
  - targetResource : Resources::LoadingResource

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
- C -

- Code: `Models::ModelNodeType`
- Creator: `Core::Rtti`
- d -
  - DrawHandle: `Base::SkinnedMeshRendererBase`

- e -
  - EntityId: `Game::Entity`

- f -
  - FuncPtr: `Jobs::SerialJobFuncDesc`, `Jobs::TPJobFuncDesc`

- i -
  - Id: `InternalGraphics::InternalGraphicsEntity`
  - Iterator: `Util::Array< TYPE >`, `Util::FixedArray< TYPE >`, `Util::Array< TYPE >`

- m -
  - Mask: `CoreGraphics::ShaderFeature`

- n -
  - Name: `Base::ShaderVariableBase`, `Models::ModelNodeType`, `CoreGraphics::ShaderFeature`

- p -
  - Position: `IO::Stream`

- s -
  - Semantic: `Base::ShaderVariableBase`

- y -
Year: Base::CalendarTimeBase

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
- a -

- **AccessMode** : `IO::Stream`
- **AccessPattern** : `IO::Stream`
- Axis: Base::GamePadBase

- b -
  - Button: Base::GamePadBase

- c -
  - CallbackType: Game::Property
  - ClearFlag: Base::RenderTargetBase
  - CpuType: Base::SystemInfoBase
  - CubeFace: Base::TextureBase

- e -
  - ErrorCode: Win360::Win360Socket

- l -
  - LinkType: InternalGraphics::InternalGraphicsEntity

- m -
  - Month: Base::CalendarTimeBase

- p -
  - Platform: Base::SystemInfoBase
  - Priority: Win360::Win360Thread, Resources::ManagedResource, Win360::Win360Thread, Resources::ManagedResource
  - Protocol: Win360::Win360Socket

- s -
  - SeekOrigin: IO::Stream
  - State: Resources::Resource
- t -
  - Type: Base::ShaderVariableBase, Base::TextureBase, Base::ShaderVariableBase

- u -
  - Usage: Base::ResourceBase

- w -
  - Weekday: Base::CalendarTimeBase

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
operator!= : OSX::OSXFileTime, Animation::AnimEventInfo, Win360::Win360FileTime, Util::String, CoreAnimation::AnimEvent
operator< : OSX::OSXFileTime, Win360::Win360FileTime, Animation::AnimEventInfo, CoreAnimation::AnimEvent, Util::String, Resources::LoadingResource
operator<= : Animation::AnimEventInfo, CoreAnimation::AnimEvent, Util::String
operator== : OSX::OSXFileTime, Animation::AnimEventInfo, Util::String, Win360::Win360FileTime, Util::String, CoreAnimation::AnimEvent
operator> : Animation::AnimEventInfo, Win360::Win360FileTime, CoreAnimation::AnimEvent, OSX::OSXFileTime, Util::String
operator> = : Util::String, Animation::AnimEventInfo, CoreAnimation::AnimEvent
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Enumerator
- Related Functions

- _
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- a -

- a() : Math::plane
- abs() : Math::float2, Math::float4
- AbstractLightEntity() : Graphics::AbstractLightEntity
- Accept() : Win360::Win360Socket
- AcceptsMessage() : Messaging::Port, Game::Entity, Messaging::Port
- AccessMode : IO::Stream
- AccessPattern : IO::Stream
- AcquireJointTextureRowPointer() : Base::SkinnedMeshRendererBase
- Action() : Actions::Action
- ActionReader() : Script::ActionReader
- ActivateDialog() : Script::Dialog, Script::DialogManager
- ActivateEntity() : BaseGameFeature::EntityManager
- ActorPhysicsProperty() : PhysicsFeature::ActorPhysicsProperty
- AdapterExists() : Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice, Graphics::Display, Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice
- AdapterInfo() : CoreGraphics::AdapterInfo
- Add() : Util::Dictionary< KEYTYPE, VALUETYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::RingBuffer< TYPE >, Models::VisResolveContainer< TYPE >,
Win360::Win360Interlocked, OSX::OSXInterlocked
- AddAction() : Script::DialogTake, FSM::Transition
- AddAfter() : Util::List< TYPE >
- AddAttr() : Http::HtmlPageWriter
- AddBack() : Util::List< TYPE >
- AddBatch() : Frame::FramePassBase
- AddBefore() : Util::List< TYPE >
- AddBlob() : Game::Entity
- AddBool() : Game::Entity
- AddCategoryAttr() : BaseGameFeature::CategoryManager
- AddChild() : Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode
- AddChildTake() : Script::DialogTake
- AddCondition() : Conditions::And, Conditions::Or, FSM::Transition
- AddCurParticleVertexIndex() : Direct3D9::D3D9ParticleRenderer
- AddDeferredMessage() : InternalGraphics::InternalGraphicsEntity
- AddDelayedJob() : BaseGameFeature::EntityManager
- AddDependency() : InternalGraphics::InternalView
- AddDepthStencilBuffer() : Base::RenderTargetBase
- AddEntryAction() : FSM::State
- AddEvent() : CoreAnimation::AnimClip
- AddExitAction() : FSM::State
- AddFloat() : Game::Entity
- AddFloat4() : Game::Entity
- AddFragment() : Characters::CharacterSkinNode
- AddFrameAction() : FSM::State
- AddFramePassBase() : Frame::FrameShader
- AddFront() : Util::List< TYPE >
- AddGraphicsAttachment() : GraphicsFeature::AttachmentManager
- AddGraphicsAttachmentTemporary() : GraphicsFeature::AttachmentManager
- AddGuid() : Game::Entity
- AddInfo() : Script::InfoLog
- AddInt() : Game::Entity
- AddLink() : InternalGraphics::InternalGraphicsEntity
- AddList() : Util::List<TYPE>
- AddMatrix44() : Game::Entity
- AddMessage() : FrameSync::FrameSyncHandlerThread, Messaging::BatchMessage, Messaging::BlockingHandlerThread, Messaging::RunThroughHandlerThread, Messaging::HandlerThreadBase
- AddMultiple() : Util::SparseTable<TYPE>
- AddMultipleRenderTarget() : Frame::FrameShader
- AddProperties() : BaseGameFeature::FactoryManager
- AddRef() : Core::RefCounted, Util::SimpleTree<VALUETYPE>::Node, Core::RefCounted
- AddReference() : Util::SparseTable<TYPE>
- AddRenderTarget() : Base::MultipleRenderTargetBase, Frame::FrameShader, Base::MultipleRenderTargetBase
- AddSample() : Math::Extrapolator<TYPE>
- AddShaderParam() : Models::StateNode
- AddShape() : Base::ShapeRendererBase
- AddShapes() : Base::ShapeRendererBase, PhysicsFeature::EnvironmentCollideProperty
- AddSharedDepthStencilBuffer() : Base::RenderTargetBase
- AddSingle() : Util::SparseTable<TYPE>
- AddSkin() : Characters::CharacterSkinLibrary, Characters::CharacterSkinSet
- AddSkinList() : Characters::CharacterSkinLibrary
- AddState() : FSM::StateMachine
- AddStateHandler() : App::GameApplication
- AddString() : Game::Entity, Util::StringBuffer
- AddText() : Script::DialogTake
- AddTextElement() : Base::TextRendererBase
- AddTextElements() : Base::TextRendererBase
- AddTexture() : Frame::FrameShader
- AddTrack() : Http::SvgLineChartWriter
- AddTrackedCharJoint() : InternalGraphics::InternalModelEntity, Graphics::ModelEntity
- AddTransition() : FSM::State
- AddVariable() : Frame::FramePassBase, Frame::FrameBatch
- AddVisibleNodeInstance() : Models::ModelNode
- AddVisibleParticleSystem() : Base::ParticleRendererBase
- AddWireFrameBox() : Base::ShapeRendererBase
- AdvanceProgress() : BaseGameFeature::LoaderServer
- affinetransformation() : Math::matrix44
- all() : Math::float2
- Alloc() : OSX::OSXHeap, Memory::PoolArrayAllocator, OSX::OSXMemoryPool, Win360::Win360Heap, Win360::Win360MemoryPool
- AllocInstanceMemory() : Core::Rtti
- AllocJointTextureRow() : Base::SkinnedMeshRendererBase
- AllocPrivateBuffer() : Base::JobBase
- And() : Util::BitField< NUMBITS >
- angle() : Math::float4
- AnimationLibrary() : Characters::Character
- AnimClip() : CoreAnimation::AnimClip
- AnimController() : Characters::CharacterInstance
- AnimCurve() : CoreAnimation::AnimCurve
- AnimEvent() : CoreAnimation::AnimEvent
- AnimEventHandlerBase() : Animation::AnimEventHandlerBase
- AnimEventManager() : BaseGameFeature::AnimEventManager
- AnimEventServer() : Animation::AnimEventServer
- AnimJob() : Animation::AnimJob
- AnimKeyBuffer() : CoreAnimation::AnimKeyBuffer
- AnimResource() : CoreAnimation::AnimResource
- AnimSampleBuffer() : CoreAnimation::AnimSampleBuffer
- AnimSequencer() : Characters::CharacterAnimationController, Animation::AnimSequencer
- any() : Math::float2
- Append() : Util::Array< TYPE >, Util::String, Util::SimpleTree< VALUETYPE >::Node, Util::String, Util::Array< TYPE >
- AppendAction() : Actions::SequenceAction
- AppendArray() : Util::Array< TYPE >
- AppendBool() : Util::String
- AppendCommandString() : Util::CommandLineArgs
- AppendFloat() : Util::String
- AppendFloat2() : Util::String
- AppendFloat4() : Util::String
- `AppendInt()` : `Util::String`
- `AppendLoadingResource()` : `Resources::ResourceScheduler`
- `AppendLocalPath()` : `IO::URI`
- `AppendMatrix44()` : `Util::String`
- `AppendRange()` : `Util::String`
- `Application()` : `App::Application`
- `Apply()` : `Base::ShaderVariableInstanceBase`
- `ApplyCameraSettings()` : `InternalGraphics::InternalView`
- `ApplyGlobalAudioPitch()` : `BaseGameFeature::TimeManager`
- `ApplyImpulseAtPos()` : `PhysicsFeature::PhysicsProperty`
- `ApplyJointComponents()` :
  - `Characters::CharacterSkeletonInstance`
- `ApplyModelEntityLights()` : `Lighting::LightServerBase`, `Lighting::SM30LightServer`, `Lighting::LightServerBase`
- `ApplyModelTransforms()` : `Win360::D3D9TransformDevice`, `Base::TransformDeviceBase`, `Win360::D3D9TransformDevice`
- `ApplyObjectId()` : `Base::ShaderServerBase`
- `ApplyPrimitives()` : `Base::MeshBase`
- `ApplySharedState()` :
  - `Models::ModelNode`, `Models::ShapeNode`, `Models::ModelNode`, `Characters::CharacterSkinNode`, `Models::StateNode`, `Models::ModelNode`, `Models::StateNode`
- `ApplySkinList()` : `Characters::CharacterSkinSet`
- `ApplyState()` : `Models::StateNodeInstance`, `Models::TransformNodeInstance`, `Models::ModelNodeInstance`, `Particles::ParticleSystemNodeInstance`, `Models::TransformNodeInstance`
- `ApplyTimeEffect()` : `BaseGameFeature::TimeManager`
- `ApplyViewSettings()` : `Win360::D3D9TransformDevice`, `Base::TransformDeviceBase`, `Win360::D3D9TransformDevice`
- `ArchiveBase()` : `IO::ArchiveBase`
- `ArchiveFileSystem()` : `IO::ArchiveFileSystem`
- `AreKeySliceValuesValid()` : `CoreAnimation::AnimClip`
- `AreMipMapsEnabled()` : `Base::RenderTargetBase`
- `AreSubTasksLoaded()` : `Script::Task`
- `Array()` : `Util::Array<TYPE>`
- `Arrive()` : `Win360::Win360ThreadBarrier`
- `ArriveAtSyncPoint()` : `FrameSync::FrameSyncHandlerThread`
- As() : Util::Array< TYPE >
- as() : Math::bbox
- As() : Util::String
- as() : Math::matrix44, Math::float2, Math::frustum
- As() : Util::Array< TYPE >
- AsArray() : Util::FixedArray< TYPE >, Util::RingBuffer< TYPE >
- AsBinary() : OSX::OSXGuid, Win32::Win32Guid
- AsBool() : Util::String
- AsCharPtr() : Util::String
- AsD3D9MultiSampleType() : Win360::D3D9Types
- AsD3D9PixelFormat() : Win360::D3D9Types
- AsD3D9Pool() : Win360::D3D9Types
- AsD3D9PrimitiveType() : Win360::D3D9Types
- AsD3D9Usage() : Win360::D3D9Types
- AsD3D9VertexDeclarationType() : Win360::D3D9Types
- AsD3D9VertexDeclarationUsage() : Win360::D3D9Types
- AsD3DXImageFileFormat() : Win360::D3D9Types
- AsFloat() : Util::String
- AsFloat2() : Util::String
- AsFloat4() : Util::String
- AsInt() : Util::String
- AsMatrix44() : Util::String
- AsNebulaPixelFormat() : Win360::D3D9Types
- AsRootKey() : System::Win32Registry
- AsSampleIndex() : Particles::EnvelopeSampleBuffer
- Assert() : Conditions::And, Conditions::Condition, Actions::Action, Conditions::Condition, Script::Task, Script::DialogTake, Conditions::And, Conditions::Or, Conditions::Not, Conditions::Condition, Actions::ActionList, Actions::Action, Actions::IfThenElseAction, Actions::Action, Actions::IfThenElseAction, Actions::SequenceAction, Conditions::Not, Actions::ActionList, Conditions::Or, Conditions::Condition
- AssertDialog() : Script::Dialog
- AssertDialogTakeExists() : Script::DialogManager
- Assign() : IO::Assign
- AssignRegistry() : IO::AssignRegistry
- AssignRenderBufferTextures() : Lighting::LightPrePassServer
- AsString() : Util::FourCC, Util::StringAtom, OSX::OSXGuid, IO::URI, Win360::Win360FileTime, IO::MediaType, Win32::Win32Guid
- AsUInt() : Util::FourCC
- AsyncPort() : Messaging::AsyncPort
- At() : Util::FixedTable< TYPE >
- AttachDisplayEventHandler() : Graphics::Display
- AttachEntity() : InternalGraphics::AttachmentServer, BaseGameFeature::EntityManager, InternalGraphics::InternalStage
- AttachEntityLoader() : BaseGameFeature::LoaderServer
- AttachEntityTemporary() : InternalGraphics::AttachmentServer
- AttachEventHandler() : Base::DisplayDeviceBase, Base::RenderDeviceBase, Base::DisplayDeviceBase
- AttachGameFeature() : Game::GameServer
- AttachHandler() : Messaging::Port, Interface::InterfaceBase, Messaging::Port, Interface::InterfaceBase, Messaging::Port, Messaging::HandlerThreadBase, Interface::InterfaceBase, Messaging::Port, Messaging::HandlerThreadBase, Messaging::Port, Messaging::AsyncPort, Messaging::Port, Interface::InterfaceBase, Messaging::Port, IO::Console, Messaging::HandlerThreadBase, Messaging::Port, Messaging::HandlerThreadBase, Messaging::Port
- AttachInputHandler() : Base::InputServerBase
- AttachManager() : Game::FeatureUnit
- AttachMapper() : Resources::ResourceManager
- AttachmentManager() : GraphicsFeature::AttachmentManager
- AttachmentServer() : InternalGraphics::AttachmentServer
- AttachNode() : Models::Model
- AttachNodeInstance() : Models::ModelInstance
- AttachPort() : Messaging::Dispatcher
- AttachRenderEventHandler() : Graphics::Display
- AttachRequestHandler() : Http::HttpServer, Http::HttpServerProxy
- AttachTimeSource() : BaseGameFeature::TimeManager
- AttachVisibilityContainer() : Visibility::VisibilityChecker
- AttachVisibilitySystem() : Visibility::VisibilityChecker,
Visibility::VisibilityQuery
- AttachVisibilitySystems() : InternalGraphics::InternalStage, Visibility::VisibilityChecker
- AttachVisibleLight() : Lighting::LightServerBase, Lighting::LightPrePassServer
- AttachVisibleModelInstance() : Models::VisResolver
- AttachVisibleModelInstancePlayerCamera() : Models::VisResolver
- AutoEvade() : PhysicsFeature::ActorPhysicsProperty
- AutoEvadeProbeAboveGround : PhysicsFeature::ActorPhysicsProperty
- AutoEvadeProbeRadius : PhysicsFeature::ActorPhysicsProperty
- AutoManageManagedResource() : Resources::ResourceManager
- Axis : Base::GamePadBase
- AxisAsString() : Base::GamePadBase
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q  
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **b** -

  - b() : Math::plane
  - Back() : Util::Array< TYPE >, Util::List< TYPE >, Util::RingBuffer< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::Array< TYPE >, Util::List< TYPE >
  - barycentric() : Math::float4, Math::quaternion
  - BatchMessage() : Messaging::BatchMessage
  - bbox() : Math::bbox
  - Begin() : Direct3D9::D3D9ShaderInstance, Util::Crc, Util::FixedArray< TYPE >, Util::Array< TYPE >, Util::List< TYPE >, Http::HtmlPageWriter, Util::List< TYPE >, Direct3D9::D3D9ShaderInstance, Util::Array< TYPE >, Debug::DebugCounter, Base::ShaderInstanceBase
  - begin_extend() : Math::bbox
  - BeginAddCategoryAttrs() : BaseGameFeature::CategoryManager
  - BeginAttach() : Direct3D9::D3D9ParticleRenderer, Base::ParticleRendererBase, Direct3D9::D3D9ParticleRenderer
  - BeginAttachVisibilityContainer() : Visibility::VisibilitySystemBase, Visibility::VisibilityChecker, Visibility::VisibilitySystemBase
  - BeginAttachVisibleLights() : Lighting::LightServerBase
  - BeginBatch() : Base::MultipleRenderTargetBase,
Base::RenderDeviceBase, Base::RenderTargetBase, Base::RenderDeviceBase, Base::RenderTargetBase
- BeginBulkAdd() : Util::Dictionary<KEYTYPE, VALUETYPE>
- BeginCapture() : Input::InputHandler, Base::KeyboardBase, Base::MouseBase, Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- BeginDraw() : Characters::CharacterServer
- BeginEvents() : CoreAnimation::AnimClip
- BeginFrame() : Direct3D9::D3D9RenderDevice, Base::InputServerBase, Lighting::LightServerBase, Base::InputServerBase, Lighting::LightServerBase, Base::InputServerBase, Lighting::LightServerBase, Characters::CharacterServer, Direct3D9::D3D9RenderDevice, Base::RenderDeviceBase
- BeginGather() : Characters::CharacterServer
- BeginGatherSkins() : Base::SkinnedMeshRendererBase
- BEGINLABEL : Script::InfoLog
- BeginNode() : Http::SvgPageWriter, IO::XmlWriter, Http::SvgPageWriter
- BeginPaintGroup() : Http::SvgPageWriter
- BeginParseDataTags() : Models::ModelNode
- BeginPass() : Base::RenderDeviceBase, Direct3D9::D3D9ShaderInstance, Direct3D9::D3D9RenderTarget, Base::MultipleRenderTargetBase, Base::RenderDeviceBase, Base::MultipleRenderTargetBase, Base::RenderDeviceBase, Direct3D9::D3D9RenderTarget, Base::RenderDeviceBase, Base::RenderTargetBase, Base::ShaderInstanceBase
- BeginResolve() : Models::VisResolver
- BeginSection() : Script::InfoLog
- BeginSetup() : Util::SparseTable<TYPE>
- BeginTextGroup() : Http::SvgPageWriter
- BeginTransformGroup() : Http::SvgPageWriter
- BinaryReader() : IO::BinaryReader
- BinarySearchIndex() : Util::Array<TYPE>, Util::FixedArray<TYPE>
- BinarySize() : OSX::OSXGuid, Win32::Win32Guid
- BinaryWriter() : IO::BinaryWriter
- Bind() : Base::ShaderVariableInstanceBase,
Messaging::DelegateTable, Win360::Win360Socket, Base::ShaderVariableInstanceBase
- BitField() : Util::BitField< NUMBITS >
- BitsPerPixel() : Resources::D3D9TextureStreamer
- Blob() : Util::Blob
- BlockingHandlerThread() : Messaging::BlockingHandlerThread
- Broadcast() : Net::StdTcpServer
- BuildSignature() : Base::VertexLayoutBase
- BuildStateMachineRacks() : Script::ScriptManager
- Button : Base::GamePadBase
- ButtonAsString() : Base::GamePadBase
- ButtonDoubleClicked() : Base::MouseBase
- ButtonDown() : Base::MouseBase, Base::GamePadBase, Base::MouseBase, Base::GamePadBase
- ButtonPressed() : Base::GamePadBase, Base::MouseBase, Base::GamePadBase, Base::MouseBase
- ButtonUp() : Base::GamePadBase, Base::MouseBase, Base::GamePadBase, Base::MouseBase
- BXmlLoaderUtil() : IO::BXmlLoaderUtil
- BXmlReader() : IO::BXmlReader
- ByteOrder() : System::ByteOrder
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- C -

- c() : Math::plane
- CalendarTimeBase() : Base::CalendarTimeBase
- CallbackType : Game::Property
- CameraEntity() : Graphics::CameraEntity
- CameraProperty() : GraphicsFeature::CameraProperty
- CameraSettings() : Shared::CameraSettings
- CanBeMapped() : IO::Stream, IO::ZipFileStream, IO::FileStream, IO::MemoryStream
- Cancel() : Messaging::AsyncPort
- CancelCurrentDialog() : Script::DialogManager
- CancelMessage() : FrameSync::FrameSyncHandlerThread, Messaging::BlockingHandlerThread, Messaging::HandlerThreadBase, Messaging::RunThroughHandlerThread
- CanCreate() : Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
- CanLoadAsync() : Models::StreamModelLoader, Resources::ResourceLoader, Resources::StreamResourceLoader, Resources::ResourceLoader, Direct3D9::D3D9StreamShaderLoader, Resources::StreamResourceLoader, Direct3D9::D3D9StreamShaderLoader,
Resources::StreamResourceLoader
- CanRead() : IO::Stream, IO::FileStream, IO::ZipFileStream, IO::MemoryStream
- CanSeek() : IO::Stream, IO::ZipFileStream, IO::FileStream, IO::MemoryStream
- CanWrite() : IO::Stream, IO::ZipFileStream, IO::FileStream, IO::MemoryStream
- Capacity() : Util::Array<TYPE>, Util::HashTable<KEYTYPE, VALUETYPE>, Util::Array<TYPE>, Util::RingBuffer<TYPE>
- cast() : Ptr<TYPE>
- Category() : BaseGameFeature::CategoryManager::Category, BaseGameFeature::CategoryManager::Entry
- CategoryManager() : BaseGameFeature::CategoryManager
- catmullrom() : Math::float4
- center() : Math::bbox
- centerX() : Math::rectangle<TYPE>
- centerY() : Math::rectangle<TYPE>
- ChangeFileExtension() : Util::String
- Character() : Characters::Character
- CharacterAnimationController() : Characters::CharacterAnimationController
- CharacterInstance() : Characters::CharacterInstance
- CharacterJoint() : Characters::CharacterJoint
- CharacterNode() : Characters::CharacterNode
- CharacterNodeInstance() : Characters::CharacterNodeInstance
- CharacterServer() : Characters::CharacterServer
- CharacterSkeleton() : Characters::CharacterSkeleton
- CharacterSkeletonInstance() : Characters::CharacterSkeletonInstance
- CharacterSkin() : Characters::CharacterSkin
- CharacterSkinLibrary() : Characters::CharacterSkinLibrary
- CharacterSkinList() : Characters::CharacterSkinList
- CharacterSkinNode() : Characters::CharacterSkinNode
- CharacterSkinNodeInstance() : Characters::CharacterSkinNodeInstance
- CharacterSkinSet() : Characters::CharacterSkinSet
- CharacterVariationSet() : Characters::CharacterVariationSet
- CharJointInfo() : Shared::CharJointInfo
- ChaseCameraProperty()
  - GraphicsFeature::ChaseCameraProperty
- CheckDebugRendering() : Game::GameServer
- CheckDone() : Jobs::TPJobPort, Jobs::SerialJobPort, Base::JobPortBase, Jobs::TPJobPort
- CheckFileExtension() : Util::String
- CheckId() : Messaging::Message
- CheckLodDistance() : Models::TransformNode
- CheckPendingResources() : Models::Model, Resources::ResourceManager
- CheckUpdateDone() : Characters::CharacterInstance
- CheckValidCharSet() : Util::String
- Child() : Util::SimpleTree< VALUETYPE >::Node
- ChooseResponse() : Script::DialogManager
- Circle() : Http::SvgPageWriter
- clamp() : Math::float4
- ClampKeyIndex() : CoreAnimation::AnimUtil
- ClassExists() : Core::Factory
- Cleanup() : BaseGameFeature::EntityManager, Base::ShaderInstanceBase, Direct3D9::D3D9ShaderInstance
- CleanupGameFeatures() : App::GameApplication
- CleanupLoadingQueue() : Resources::ResourceScheduler
- CleanupStateHandlers() : App::GameApplication
- Clear() : Util::SparseTable< TYPE >, Threading::SafeFlag, Threading::SafePriorityQueue< PRITYPE, TYPE >, Threading::SafeQueue< TYPE >, Util::Array< TYPE >, Util::BitField< NUMBITS >, Resources::ManagedResource, IO::URI, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::FixedArray< TYPE >, Util::FixedTable< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::List< TYPE >, Util::Array< TYPE >, Resources::ManagedResource, Util::HashTable< KEYTYPE, VALUETYPE >, Util::Queue< TYPE >, Resources::ManagedResource, Util::Stack< TYPE >, Util::String, Util::StringAtom, Util::Variant, Characters::CharacterSkinSet, IO::MediaType, Util::List< TYPE >, Resources::ManagedResource
- ClearAssign() : IO::AssignRegistry
- ClearAttachments() : GraphicsFeature::AttachmentManager
- ClearAttachmentsOnEntity() :
GraphicsFeature::AttachmentManager
- ClearBit() : Util::BitField< NUMBITS >
- ClearCapture() : Base::InputServerBase
- ClearEntities() : PhysicsFeature::TriggerProperty
- ClearEntity() : Game::Property
- ClearEnvEntity() : BaseGameFeature::EnvEntityManager
- ClearFeatureBits() : Base::ShaderServerBase
- ClearFlag : Base::RenderTargetBase
- ClearHandlers() : Messaging::HandlerThreadBase
- ClearKeyboardCapture() : Base::InputServerBase
- ClearLinks() : InternalGraphics::InternalGraphicsEntity
- ClearMouseCapture() : Base::InputServerBase
- ClearRenderStats() : Resources::ManagedResource
- ClearVisibilityLinks() : Visibility::VisibilityChecker
- Client() : FrameSync::FrameSyncSharedData
- ClientDiscard() : FrameSync::FrameSyncSharedData
- ClientSetup() : FrameSync::FrameSyncSharedData
- clip() : Math::frustum, Math::plane
- clipmask() : Math::frustum
- clipstatus() : Math::bbox, Math::frustum, Math::sphere
- Close() : IO::BinaryWriter, InternalGraphics::InternalGraphicsServer, IO::BinaryReader, Lighting::LightPrePassServer, IO::BXmlReader, Lighting::SM30LightServer, Direct3D9::D3D9ShaderServer, IO::ConsoleHandler, IO::Stream, Messaging::Handler, Lighting::LightServerBase, Lighting::SM30ShadowServer, IO::StreamWriter, Particles::ParticleServer, Win32::Win32InputServer, Win32::Win32DisplayDevice, Resources::ResourceManager, Models::VisResolver, IO::Console, Net::StdTcpServer, Win360::Win360Socket, IO::StreamReader, Messaging::AsyncPort, Win360::Win360Socket, Lighting::LightPrePassServer, Visibility::VisibilityChecker, Visibility::VisibilityBoxSystem, IO::XmlWriter, IO::StreamWriter, Visibility::VisibilityQuadtree, IO::XmlReader, App::GameApplication, IO::StreamWriter, IO::ZipFileEntry, Models::ModelServer, Direct3D9::D3D9RenderDevice, InternalGraphics::AttachmentServer, IO::ConsoleHandler, App::Application, IO::ZipFileStream, IO::ExcelXmlReader,
Game::GameServer, Http::SvgPageWriter,
App::ConsoleApplication, Animation::AnimEventServer,
App::RenderApplication, App::ViewerApplication,
FSM::StateMachine, Http::HtmlPageWriter,
IO::ConsoleHandler, Base::InputServerBase,
Lighting::SM30ShadowServer, Net::StdTcpServer,
Http::HttpMessageHandler, IO::FileStream,
IO::StreamReader, IO::StreamWriter, IO::IoInterfaceHandler,
IO::StreamReader, Http::HttpServer, Messaging::AsyncPort,
Graphics::GraphicsInterface, Core::CoreServer,
Base::DisplayDeviceBase, Base::RenderDeviceBase,
Base::ShaderServerBase, Http::HttpServerProxy,
IO::LogFileConsoleHandler, Base::ShapeRendererBase,
Win32::Win32DisplayDevice, Base::TextRendererBase,
Base::TransformDeviceBase,
Base::VertexLayoutServerBase,
Visibility::VisibilitySystemBase, Http::SvgPageWriter,
Direct3D9::D3D9RenderDevice, Messaging::AsyncPort,
Direct3D9::D3D9ShaderServer, Debug::DebugHandler,
Direct3D9::D3D9TextRenderer, Win32::Win32DisplayDevice,
Messaging::Handler, Messaging::AsyncPort,
IO::StreamWriter, BaseGameFeature::LoaderServer,
Visibility::VisibilityClusterSystem, Graphics::GraphicsServer,
, Direct3D9::D3D9TextRenderer,
Win360::D3D9ShapeRenderer,
Win360::D3D9TransformDevice,
Base::VertexLayoutServerBase, Frame::FrameServer,
Messaging::Handler, Messaging::AsyncPort,
Graphics::Display, IO::MemoryStream,
Graphics::GraphicsHandler, IO::ConsoleHandler,
Messaging::Handler, Win360::D3D9TransformDevice,
Win32::Win32InputServer, Win360::D3D9ShapeRenderer

- CloseDInputMouse() : Win32::Win32InputServer
- CloseFile() : OSX::OSXFSWrapper,
  Win360::Win360FSWrapper
- CloseProgressIndicator() : BaseGameFeature::LoaderServer
- closestpoint() : Math::line
- CloseTask() : Script::Task
- CloseWindow() : Win32::Win32DisplayDevice
- Code: `Models::ModelNodeType`
- Column(): `Util::QuadTree<TYPE>::Node`
- CommandLineArgs(): `Util::CommandLineArgs`
- Commit(): `Base::ShaderInstanceBase`, `Direct3D9::D3D9ShaderInstance`
- CommitChangesToDatabase(): `BaseGameFeature::CategoryManager`
- CompareExchange(): `Win360::Win360Interlocked`, `OSX::OSXInterlocked`
- Compute(): `Lighting::PSSMUtil`, `Util::Crc`
- ComputeAbsMousePos(): `Win32::Win32DisplayDevice`
- ComputeAdjustedWindowRect(): `Win32::Win32DisplayDevice`
- ComputeAlignedBlockSize(): `OSX::OSXMemoryPool`, `Win360::Win360MemoryPool`
- ComputeBlendWeight(): `Animation::AnimJob`
- ComputeDecodedSize(): `Util::RunLengthCodec`
- ComputeFileCrc(): `Io::IoServer`
- ComputeKeySlicePointerAndSize(): `CoreAnimation::AnimResource`
- ComputeMinMaxAvgCounts(): `Debug::DebugPageHandler`
- ComputeMinMaxAvgTimes(): `Debug::DebugPageHandler`
- ComputeMouseWorldRay(): `BaseGameFeature::EnvQueryManager`
- ComputeNormMousePos(): `Win32::Win32DisplayDevice`
- ComputeWorldMouseRay(): `RenderUtil::MouseRayUtil`
- Concatenate(): `Util::String`
- Condition(): `Conditions::Condition`
- ConfigureAnimDrivenMotionTracking(): `InternalGraphics::InternalModelEntity`, `Graphics::ModelEntity`
- ConfigureAnimEventTracking(): `InternalGraphics::InternalModelEntity`,
Graphics::ModelEntity
- ConfigureCharJointTracking() : Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- Confirm() : OSX::OSXConsoleHandler, IO::ConsoleHandler, IO::Console, Win32::Win32ConsoleHandler, IO::Console, IO::ConsoleHandler
- conjugate() : Math::quaternion
- Connect() : Net::StdTcpClient, Net::StdTcpClientConnection, Net::StdTcpClient, Net::MessageClient, Win360::Win360Socket, Net::MessageClientConnection
- Console() : IO::Console
- ConsoleApplication() : App::ConsoleApplication
- ConsoleHandler() : IO::ConsoleHandler
- ConsolePageHandler() : Debug::ConsolePageHandler
- ConstructContent() : Script::ScriptManager
- ConsumeNewModelNodeInstanceIndex() : Models::ModelServer
- contains() : Math::bbox
- Contains() : Util::Queue< TYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::Stack< TYPE >
- contains() : Math::bbox
- Contains() : Util::Queue< TYPE >
- ContainsCharFromSet() : Util::String
- ContainsDescription() : Script::InfoLogFilter
- ContainsInformation() : Script::InfoLogFilter
- ContainsLogLevel() : Script::InfoLogFilter
- ContainsSource() : Script::InfoLogFilter
- Content() : Util::HashTable< KEYTYPE, VALUETYPE >
- Continue() : BaseGameFeature::TimeSource
- ContinueAll() : BaseGameFeature::TimeManager
- ContinueFollow() : PhysicsFeature::ActorPhysicsProperty
- ContinueGoto() : PhysicsFeature::ActorPhysicsProperty
- Convert() : System::ByteOrder
- ConvertBackslashes() : Util::String
- ConvertInPlace() : System::ByteOrder
- ConvertPath() : OSX::OSXFSWrapper
- ConvertToArchiveURI() : IO::ZipArchive, IO::ArchiveBase
- ConvertToPathInRangeArchive() : IO::ZipArchive, IO::ArchiveBase
- CopyFile() : IO::IoServer
- CopyToBuffer() : Util::String
- CorePageHandler() : Debug::CorePageHandler
- CoreServer() : Core::CoreServer
- corner_point() : Math::bbox
- CpuType : Base::SystemInfoBase
- CpuTypeAsString() : Base::SystemInfoBase
- Cr() : Util::Crc
- Create() : Core::Rtti, Core::Factory
- CreateActionFromString() : Actions::Action
- CreateActionsFromString() : Actions::Action
- CreateAndSetupGraphicsEntities() : GraphicsFeature::SegmentedGfxUtil
- CreateAsSingleton() : Debug::DebugTimer
- CreateCollisionShape() : PhysicsFeature::TriggerProperty
- CreateConditionFromString() : Conditions::Condition
- CreateConditionsFromString() : Conditions::Condition
- CreateDialog() : Script::DialogManager
- CreateDialogTake() : Script::Dialog
- CreateDirectory() : OSX::OSXFSWrapper, IO::IoServer, Win360::Win360FSWrapper
- CreateDummyInstance() : BaseGameFeature::CategoryManager
- CreateEntityByAttrs() : BaseGameFeature::FactoryManager
- CreateEntityByCategory() : BaseGameFeature::FactoryManager
- CreateEntityByClassName() : BaseGameFeature::FactoryManager
- CreateEntityByEntity() : BaseGameFeature::FactoryManager
- CreateEntityByEntityAsCategory() : BaseGameFeature::FactoryManager
- CreateEntityByGuid() : BaseGameFeature::FactoryManager
- CreateEntityByKeyAttr() : BaseGameFeature::FactoryManager
- CreateEntityByTemplate() : BaseGameFeature::FactoryManager
- CreateEntityByTemplateAsCategory() : BaseGameFeature::FactoryManager
- CreateEntityCommand() : Commands::CreateEntityCommand
- CreateEnvEntity() : BaseGameFeature::EnvEntityManager
- CreateEvaluationJob() : Animation::AnimJob
- CreateInstance() : Base::ShaderVariableBase, Models::Model
- CreateInstanceFromAttrs() : BaseGameFeature::CategoryManager
- CreateInstanceFromInstance() : BaseGameFeature::CategoryManager
- CreateInstanceFromInstanceAsCategory() : BaseGameFeature::CategoryManager
- CreateInstanceFromTemplate() : BaseGameFeature::CategoryManager
- CreateInstanceFromTemplateAsCategory() : BaseGameFeature::CategoryManager
- CreateManagedResource() : Resources::ResourceManager
- CreateNodeInstance() : Particles::ParticleSystemNode, Characters::CharacterSkinNode, Models::ShapeNode, Models::StateNode, Models::TransformNode, Models::ModelNode, Characters::CharacterNode
- CreateNodeInstanceHierarchy() : Models::ModelNode
- CreatePartialInstance() : Models::Model
- CreatePhysicsEntity() : PhysicsFeature::ActorPhysicsProperty
- CreateProperty() : BaseGameFeature::FactoryManager
- CreateSampleAndMixJob() : CoreAnimation::AnimUtil
- CreateSampleJob() : CoreAnimation::AnimUtil
- CreateShaderInstance() : Base::ShaderBase, Base::ShaderServerBase
- CreateShaderVariableInstance() : Models::StateNodeInstance
- CreateSharedVertexLayout() : Base::VertexLayoutServerBase
- CreateStream() : IO::IoServer
- CreateUserProfile() : BaseGameFeature::LoaderServer
- CreateVisibilityJob() : Visibility::VisibilityClusterSystem, Visibility::VisibilitySystemBase, Visibility::VisibilityBoxSystem, Visibility::VisibilityQuadtrees
- Creator : Core::Rtti
- cross3() : Math::float4
- CubeFace : Base::TextureBase
CurveByIndex() : CoreAnimation::AnimClip

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
  a  
  b  
  c  
  d  
  e  
  f  
  g  
  h  
  i  
  j  
  k  
  l  
  m  
  n  
  o  
  p  
  q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **d** -

  - d() : `Math::plane`
  - D3D9DisplayDevice() : `Direct3D9::D3D9DisplayDevice`
  - D3D9IndexBuffer() : `Win360::D3D9IndexBuffer`
  - D3D9ParticleRenderer() : `Direct3D9::D3D9ParticleRenderer`
  - D3D9ParticleSystemInstance() : `Direct3D9::D3D9ParticleSystemInstance`
  - D3D9RenderDevice() : `Direct3D9::D3D9RenderDevice`
  - D3D9RenderTarget() : `Direct3D9::D3D9RenderTarget`
  - D3D9Shader() : `Direct3D9::D3D9Shader`
  - D3D9ShaderInstance() : `Direct3D9::D3D9ShaderInstance`
  - D3D9ShaderServer() : `Direct3D9::D3D9ShaderServer`
  - D3D9ShaderVariable() : `Direct3D9::D3D9ShaderVariable`
  - D3D9ShaderVariation() : `Direct3D9::D3D9ShaderVariation`
  - D3D9ShapeRenderer() : `Win360::D3D9ShapeRenderer`
  - D3D9StreamMeshLoader() : `Win360::D3D9StreamMeshLoader`
  - D3D9TextRenderer() : `Direct3D9::D3D9TextRenderer`
  - D3D9Texture() : `Direct3D9::D3D9Texture`
  - D3D9TransformDevice() : `Win360::D3D9TransformDevice`
  - D3D9VertexBuffer() : `Win360::D3D9VertexBuffer`
  - D3D9VertexLayout() : `Win360::D3D9VertexLayout`
  - DeactivateDialog() : `Script::Dialog`
  - DeactivateEntity() : `BaseGameFeature::EntityManager`
  - Debug : `Script::InfoLog`
- DebugCounter() : Debug::DebugCounter
- DebugGraphicsHandler() : Debug::DebugGraphicsHandler
- DebugHandler() : Debug::DebugHandler
- DebugInterface() : Debug::DebugInterface
- DebugMessage() : Net::DebugMessage
- DebugOut() : IO::HistoryConsoleHandler, Win32::SysFunc, OSX::OSXConsoleHandler, IO::LogFileConsoleHandler, OSX::SysFunc, Win32::Win32ConsoleHandler, IO::Console, IO::ConsoleHandler, IO::Console
- DebugPacket() : Net::DebugPacket
- DebugPageHandler() : Debug::DebugPageHandler
- DebugServer() : Debug::DebugServer
- DebugShapeRenderer() : Debug::DebugShapeRenderer
- DebugTextRenderer() : Debug::DebugTextRenderer
- DebugTimer() : Debug::DebugTimer
- Decode() : Messaging::Message, Util::RunLengthCodec, Messaging::Message
- DecodePackets() : Net::DebugPacket
- DecodeStream() : Net::TcpMessageCodec
- decompose() : Math::matrix44
- Decr() : Debug::DebugCounter
- DecrClientCount() : Resources::ManagedResource
- Decrement() : OSX::OSXInterlocked, Win360::Win360Interlocked
- DecrUseCount() : Resources::Resource
- DefaultStepTime : Particles::ParticleSystem
- DeferredHanded() : Messaging::Message
- DelayedJob() : BaseGameFeature::EntityManager::DelayedJob
- Delegate() : Util::Delegate< ARGTYPE >
- Delete() : System::Win32Registry
- DeleteDirectory() : IO::IoServer, OSX::OSXFSWrapper, Win360::Win360FSWrapper
- DeleteEntity() : BaseGameFeature::EntityManager
- DeleteEntityImmediate() : BaseGameFeature::EntityManager
- DeleteEnvEntity() : BaseGameFeature::EnvEntityManager
- DeleteFile() : IO::IoServer, OSX::OSXFSWrapper, Win360::Win360FSWrapper
- DeleteInstance() : BaseGameFeature::CategoryManager
- DeleteLevel() : `BaseGameFeature::CategoryManager`
- DeleteProfile() : `BaseGameFeature::UserProfile`
- DeleteShapes() : `PhysicsFeature::EnvironmentCollideProperty`
- DeleteShapesByThreadId() : `Base::ShapeRendererBase`
- DeleteTextElementsByThreadId() : `Base::TextRendererBase`
- Dequeue() : `Threading::SafeQueue<TYPE>`, `Threading::SafePriorityQueue<PRITYPE, TYPE>`, `Util::Queue<TYPE>`
- DequeueAll() : `Threading::SafeQueue<TYPE>`
- DequeueMessages() : `Net::TcpMessageCodec`
- Destroy() : `Core::Factory`
- DestroyCollisionShape() : `PhysicsFeature::TriggerProperty`
- DestroySingleton() : `Debug::DebugTimer`
- DetachEntity() : `InternalGraphics::AttachmentServer`
- determinant() : `Math::matrix44`
- diagonal_size() : `Math::bbox`
- Dialog() : `Script::Dialog`
- DialogDesc() : `Script::DialogDesc`
- DialogExistsById() : `Script::DialogManager`
- DialogManager() : `Script::DialogManager`
- DialogTake() : `Script::DialogTake`
- DialogTakeExistsById() : `Script::DialogManager`
- Dictionary() : `Util::Dictionary<KEYTYPE, VALUETYPE>`
- Difference() : `Util::Array<TYPE>`
- DirectoryExists() : `IO::IoServer`, `OSX::OSXFSWrapper`, `Win360::Win360FSWrapper`
- DisablePhysics() : `PhysicsFeature::PhysicsProperty`, `PhysicsFeature::ActorPhysicsProperty`
- Discard() : `Base::VertexLayoutBase`, `Direct3D9::D3D9RenderTarget`, `Graphics::GraphicsEntity`, `Debug::DebugTimer`, `Graphics::GraphicsEntity`, `Win360::D3D9VertexLayout`, `IO::SchemeRegistry`, `Characters::CharacterSkinNodeInstance`, `Models::StateNodeInstance`, `IO::BXmlLoaderUtil`, `Base::SkinnedMeshRendererBase`, `Frame::FrameBatch`, `Base::SkinnedMeshRendererBase`, `Frame::FramePass`, `Frame::FramePassBase`, `Frame::FramePostEffect`, `FrameSync::FrameSyncTimer`, `Debug::DebugCounter`, ...
Frame::FrameShader, Graphics::CameraEntity, Graphics::GraphicsEntity, IO::ZipArchive, Util::StringBuffer, Base::JobBase, Animation::AnimSequencer, IO::ZipArchive, IO::ZipFileSystem, Base::SkinnedMeshRendererBase, IO::ZipFileSystem, Models::ModellInstance, Base::GameContentServerBase, Models::ModelNodeInstance, Jobs::TPJob, Models::StateNodeInstance, Jobs::TPJobPort, Characters::Character, Jobs::TPJobSystem, Base::JobBase, Characters::CharacterAnimationController, Models::TransformNodeInstance, Base::JobPortBase, Base::MouseRenderDeviceBase, Base::ParticleRendererBase, Characters::CharacterInstance, Base::JobPortBase, Jobs::SerialJobSystem, Jobs::TPJob, Characters::CharacterNodeInstance, Particles::EmitterMesh, Models::ModelNodeInstance, Direct3D9::D3D9ParticleRenderer, Particles::EnvelopeSampleBuffer, Direct3D9::D3D9RenderTarget, IO::ArchiveBase, Characters::CharacterServer, Particles::ParticleSystem, Jobs::TPJobPort, Particles::ParticleSystemNodeInstance, RenderModules::RenderModule, RenderModules::RTPluginRegistry, Base::ShaderInstanceBase, RenderUtil::DrawFullScreenQuad, Base::ShaderInstanceBase, Direct3D9::D3D9ParticleRenderer, Characters::CharacterSkeleton, Jobs::TPJobSystem, IO::AssignRegistry, Characters::CharacterSkeletonInstance, Jobs::TPJobThreadPool, Characters::CharacterSkinLibrary, Characters::CharacterSkinSet, Characters::CharacterVariationSet, CoreAnimation::AnimKeyBuffer, CoreAnimation::AnimSampleBuffer, Base::GameContentServerBase, Base::MouseRenderDeviceBase, Base::RenderTargetBase, Win360::D3D9VertexLayout, Base::ShaderInstanceBase

- **DiscardAllStages()**: Graphics::GraphicsServer, InternalGraphics::InternalGraphicsServer
- **DiscardAllViews()**: `Graphics::GraphicsServer`, `InternalGraphics::InternalGraphicsServer`
- **DiscardHierarchy()**: `Models::ModelNodeInstance`
- **DiscardInstance()**: `Models::Model`
- **DiscardManagedModel()**: `Models::ModelServer`
- **DiscardManagedResource()**: `Resources::ResourceManager`
- **DiscardShaderInstance()**: `Base::ShaderBase`
- **DiscardStage()**: `Graphics::GraphicsServer`, `InternalGraphics::InternalGraphicsServer`
- **Disconnect()**: `Net::MessageClient`, `Net::StdTcpClient`
- **Dispatch()**: `Messaging::StaticMessageHandler`
- **Dispatcher()**: `Messaging::Dispatcher`
- **Display()**: `Graphics::Display`
- **DisplayDevice()**: `CoreGraphics::DisplayDevice`
- **DisplayDeviceBase()**: `Base::DisplayDeviceBase`
- **DisplayEvent()**: `CoreGraphics::DisplayEvent`
- **DisplayEventHandler()**: `CoreGraphics::DisplayEventHandler`
- **DisplayMode()**: `CoreGraphics::DisplayMode`, `Graphics::DisplaySettings`
- **DisplayPageHandler()**: `Debug::DisplayPageHandler`
- **DisplaySettings()**: `Graphics::DisplaySettings`
- **distance()**: `Math::line`
- **DoCollideCheck()**: `GraphicsFeature::ChaseCameraProperty`
- **DoResourceLOD()**: `Resources::TexturePoolMapperScheduler`, `Resources::ResourceScheduler`
- **DoStateTransition()**: `App::GameApplication`
- **dot()**: `Math::plane`, `Math::quaternion`
- **dot3()**: `Math::float4`
- **downcast()**: `Ptr< TYPE >`
- **DoWork()**: `Messaging::Handler`, `Messaging::BlockingHandlerThread`, `Messaging::Handler`, `Win360::Win360Thread`, `Debug::DebugHandler`, `Messaging::RunThroughHandlerThread`, `Debug::DebugGraphicsHandler`, `OSX::OSXThread`, `FrameSync::FrameSyncHandlerThread`, `Win360::Win360Thread`, `Http::HttpMessageHandler`, `Graphics::GraphicsHandler`, `Interface::InterfaceHandlerBase`
The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- e -

- `Element()` : [Http::HtmlPageWriter](#)
- `Ellipse()` : [Http::SvgPageWriter](#)
- `EmitAnimEvents()` : [Animation::AnimJob](#), [Animation::AnimSequencer](#)
- `EmitterAttrs()` : [Particles::EmitterAttrs](#)
- `EmitterMesh()` : [Particles::EmitterMesh](#)
- `EmitWakeupSignal()` : [Jobs::TPWorkerThread](#), [Win360::Win360Thread](#), [Messaging::BlockingHandlerThread](#), [Win360::Win360Thread](#), [OSX::OSXThread](#), [Win360::Win360Thread](#)
- `EnablePhysics()` : [PhysicsFeature::ActorPhysicsProperty](#), [PhysicsFeature::PhysicsProperty](#)
- `Encode()` : [Messaging::Message](#), [Util::RunLengthCodec](#), [Messaging::Message](#)
- `EncodeStream()` : [Net::DebugPacket](#)
- `EncodeToMessage()` : [Net::TcpMessageCodec](#)
- `End()` : [Util::Array< TYPE >](#), [Util::Crc](#), [Util::List< TYPE >](#), [Util::FixedArray< TYPE >](#), [Util::Array< TYPE >](#)
- `end()` : [Math::line](#)
- `End()` : [Http::HtmlPageWriter](#), [Debug::DebugCounter](#), [Util::List< TYPE >](#), [Base::ShaderInstanceBase](#), [Direct3D9::D3D9ShaderInstance](#)
- `end_extend()` : [Math::bbox](#)
- EndAddCategoryAttrs() : BaseGameFeature::CategoryManager
- EndAttach() : Direct3D9::D3D9ParticleRenderer, Base::ParticleRendererBase, Direct3D9::D3D9ParticleRenderer
- EndAttachVisibilityContainer() : Visibility::VisibilitySystemBase, Visibility::VisibilityBoxSystem, Visibility::VisibilityChecker
- EndAttachVisibleLights() : Lighting::LightServerBase
- EndBatch() : Base::MultipleRenderTargetBase, Base::RenderDeviceBase, Base::MultipleRenderTargetBase, Base::RenderDeviceBase, Base::RenderTargetBase, Base::RenderDeviceBase
- EndBulkAdd() : Util::Dictionary<KEYTYPE, VALUETYPE>
- EndCapture() : Input::InputHandler, Base::KeyboardBase, Base::MouseBase, Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- EndDraw() : Characters::CharacterServer
- EndEvents() : CoreAnimation::AnimClip
- EndFrame() : Base::InputServerBase, Lighting::LightPrePassServer, Lighting::LightServerBase, Base::InputServerBase, Characters::CharacterServer, Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice, Base::InputServerBase, Lighting::LightServerBase, Lighting::LightPrePassServer
- EndGather() : Characters::CharacterServer
- EndGatherSkins() : Base::SkinnedMeshRendererBase
- EndGroup() : Http::SvgPageWriter
- ENDLABEL : Script::InfoLog
- EndNode() : Http::SvgPageWriter, IO::XmlWriter
- EndParseDataTags() : Models::ModelNode
- EndPass() : Base::MultipleRenderTargetBase, Base::ShaderInstanceBase, Direct3D9::D3D9RenderDevice, Direct3D9::D3D9ShaderInstance, Base::RenderDeviceBase, Base::RenderTargetBase, Base::MultipleRenderTargetBase, Direct3D9::D3D9RenderDevice, Direct3D9::D3D9RenderTarget, Direct3D9::D3D9ShaderInstance
- EndResolve() : Models::VisResolver
- EndSection() : Script::InfoLog
- EndSetup() : Util::SparseTable<TYPE>
- Enqueue() : `Util::Queue< TYPE >, Threading::SafeQueue< TYPE >`
- EnqueueAnimJob() : `Animation::AnimSequencer, Characters::CharacterAnimationController`
- EnqueueArray() : `Threading::SafeQueue< TYPE >`
- Enter() : `OSX::OSXCriticalSection, Win360::Win360CriticalSection`
- EnterLockStepMode() : `FrameSync::FrameSyncHandlerThread, Graphics::GraphicsInterface`
- Entity() : `Game::Entity`
- EntityId : `Game::Entity`
- EntityIsInActiveLayer() : `BaseGameFeature::EntityLoaderBase`
- EntityLoaderBase() : `BaseGameFeature::EntityLoaderBase`
- EntityManager() : `BaseGameFeature::EntityManager`
- Entry() : `BaseGameFeature::CategoryManager::Entry, Resources::ResourceDictionary::Entry`
- EnumProfiles() : `BaseGameFeature::UserProfile`
- EnvelopeCurve() : `Particles::EnvelopeCurve`
- EnvelopeSampleBuffer() : `Particles::EnvelopeSampleBuffer`
- EnvEntityExists() : `BaseGameFeature::EnvEntityManager`
- EnvEntityManager() : `BaseGameFeature::EnvEntityManager`
- EnvironmentCollideProperty() : `PhysicsFeature::EnvironmentCollideProperty`
- EnvQueryManager() : `BaseGameFeature::EnvQueryManager`
- Eof() : `IO::ZipFileStream, IO::StreamReader, IO::FileStream, IO::StreamReader, IO::MemoryStream, IO::StreamReader, IO::Stream, OSX::OSXFSWrapper, IO::StreamReader, Win360::Win360FSWrapper`
- equal3_all() : `Math::float4`
- equal3_any() : `Math::float4`
- equal4_all() : `Math::float4`
- equal4_any() : `Math::float4`
- Erase() : `Util::Array< TYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::SimpleTree< VALUETYPE >::Node`
- EraseAtIndex() : `Util::Dictionary< KEYTYPE, VALUETYPE >`
- EraseIndex() : `Util::Array< TYPE >`
- EraseIndexSwap() : `Util::Array< TYPE >`
- EraseMatchingElements() : `Threading::SafeQueue< TYPE >`. 
Threading::SafePriorityQueue< PRITYPE, TYPE >
- EraseSwap() : Util::Array< TYPE >
- Error() : IO::ConsoleHandler, Win32::SysFunc, IO::Console, Win32::Win32ConsoleHandler, IO::LogFileConsoleHandler, IO::HistoryConsoleHandler, OSX::OSXConsoleHandler, Script::InfoLog, OSX::SysFunc
- ErrorCode : Win360::Win360Socket
- EstimateLatency() : Math::Extrapolator< TYPE >
- EstimateUpdateTime() : Math::Extrapolator< TYPE >
- Evaluate() : Conditions::Not, Conditions::Condition, Conditions::Or, Conditions::And, Conditions::Operator< TYPE >, Conditions::Condition
- EvaluateCloseConditions() : Script::Task
- EvaluateConditions() : FSM::Transition
- EvaluateFailConditions() : Script::Task
- EvaluateOpenConditions() : Script::Task
- EvaluateSubTasks() : Script::Task
- EvaluateTransitions() : FSM::State
- ExcelXmlReader() : IO::ExcelXmlReader
- Exchange() : Win360::Win360Interlocked, OSX::OSXInterlocked
- Execute() : Actions::ActionList, Actions::Action, Actions::IfThenElseAction, Actions::SequenceAction
- ExecuteActions() : Script::DialogTake, FSM::Transition
- ExecuteCloseActions() : Script::Task
- ExecuteFailActions() : Script::Task
- ExecuteOpenActions() : Script::Task
- ExecuteTakeActions() : Script::DialogManager
- Exists() : System::Win32Environment, System::Win32Registry
- ExistsEntityByAttr() : BaseGameFeature::EntityManager
- ExistsEntityByUniqueId() : BaseGameFeature::EntityManager
- Exit() : App::Application, OSX::SysFunc, App::Application, Win32::SysFunc, App::Application
- ExitHandler() : Core::ExitHandler
- exp() : Math::quaternion
- extend() : Math::bbox
- extents() : Math::bbox
- ExtractDirName() : Util::String
- ExtractFileName() : `Util::String`
- ExtractFromUri() : `Win360::Win360IpAddress`
- ExtractLastDirName() : `Util::String`
- ExtractRange() : `Util::String`
- ExtractToEnd() : `Util::String`
- ExtractToLastSlash() : `Util::String`
- Extrapolator() : `Math::Extrapolator<TYPE>`
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j
k  
l  
m  
n  
o  
p  
q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- f -

- FactoryManager() : BaseGameFeature::FactoryManager
- FailTask() : Script::Task
- FeatureMaskToString() : Base::ShaderServerBase
- FeatureStringToMask() : Base::ShaderServerBase
- FeatureUnit() : Game::FeatureUnit
- FileExists() : Win360::Win360FSWrapper, IO::IoServer, OSX::OSXFSWrapper
- FileStream() : IO::FileStream
- FileTimeToLocalTime() : Base::CalendarTimeBase, Win360::Win360CalendarTime
- FileTimeToSystemTime() : Base::CalendarTimeBase, Win360::Win360CalendarTime
- Fill() : Util::Array< TYPE >, Util::FixedArray< TYPE >
- FillActionAndConditionLists() : Script::ScriptManager
- Filter() : Script::InfoLogFilter
- FilterByDescription() : Script::InfoLogFilter
- FilterByInformation() : Script::InfoLogFilter
- FilterByLogLevel() : Script::InfoLogFilter
- FilterBySource() : Script::InfoLogFilter
- FilterEntities() : PhysicsFeature::TriggerProperty
- Find() : Util::List< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::List< TYPE >, Util::StringAtomTableBase, Util::Array< TYPE >, Util::StringAtomTableBase, Util::Array<
TYPE > , Util::FixedArray< TYPE >
- FindArchiveWithDir() : IO::ZipFileSystem
- FindArchiveWithFile() : IO::ZipFileSystem
- FindAttrIndex() : IO::BXmLLoaderUtil
- FindBluePrint() : BaseGameFeature::FactoryManager
- FindCharIndex() : Util::String
- FindChildNodeIndex() : IO::BXmLLoaderUtil
- FindColumnIndex() : IO::ExcelXmlReader
- FindComponent() : Base::VertexLayoutBase
- FindContainmentNode() : Util::QuadTree< TYPE >::Node
- FindDirEntry() : IO::ZipDirEntry
- FindFileEntry() : IO::ZipDirEntry
- FindGuidIndex() : Script::ScriptManager
- FindIndex() : Util::Array< TYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::FixedArray< TYPE >
- FindInstances() : BaseGameFeature::CategoryManager
- FindNodeIndex() : IO::BXmLLoaderUtil
- FindProperty() : Game::Entity
- FindSiblingNodeIndex() : IO::BXmLLoaderUtil
- FindStateByName() : FSM::StateMachine
- FindStateHandlerByName() : App::GameApplication
- FindStringIndex() : Util::String
- FindSubTaskByGuid() : Script::Task
- FindSubTaskById() : Script::Task
- FindTemplate() : BaseGameFeature::CategoryManager
- FindTemplateByAttr() : BaseGameFeature::CategoryManager
- FirstCharToUpper() : Util::String
- FixedArray() : Util::FixedArray< TYPE >
- FixedTable() : Util::FixedTable< TYPE >
- FixFadeTimes() : Animation::AnimJob
- float2() : Math::float2
- float4() : Math::float4
- Flush() : IO::Stream, IO::FileStream, Win360::Win360FSWrapper, IO::Stream, OSX::OSXFSWrapper
- FlushBatchedMessages() : Graphics::GraphicsInterface
- FocusManager() : BaseGameFeature::FocusManager
- Format() : Util::String, Base::CalendarTimeBase
- FormatArgList() : Util::String
- FourCC() : Util::FourCC
- Fragment() : IO::URI
- FrameBatch() : Frame::FrameBatch
- FramePass() : Frame::FramePass
- FramePassBase() : Frame::FramePassBase
- FramePostEffect() : Frame::FramePostEffect
- FrameServer() : Frame::FrameServer
- FrameShader() : Frame::FrameShader
- FrameSyncHandlerThread() : FrameSync::FrameSyncHandlerThread
- FrameSyncSharedData() : FrameSync::FrameSyncSharedData
- FrameSyncTimer() : FrameSync::FrameSyncTimer
- Free() : OSX::OSXHeap, Memory::PoolArrayAllocator, Win360::Win360MemoryPool, OSX::OSXMemoryPool, Win360::Win360Heap
- FreeInstanceMemory() : Core::Rtti
- FreeJointTextureRow() : Base::SkinnedMeshRendererBase
- From() : Util::String
- FromBinary() : OSX::OSXGuid, Win32::Win32Guid
- FromBool() : Util::String
- FromFloat() : Util::String
- FromFloat2() : Util::String
- FromFloat4() : Util::String
- FromFunction() : Util::Delegate< ARGTYPE >
- FromInt() : Util::String
- FromList() : Actions::SequenceAction
- FromMatrix44() : Util::String
- FromMediaType() : CoreGraphics::ImageFileFormat
- FromMethod() : Util::Delegate< ARGTYPE >
- FromName() : Models::ModelNodeType
- FromString() : Http::HttpStatus, Input::MouseButton, CoreAnimation::SampleType, Win32::Win32Guid, Frame::SortingMode, OSX::OSXGuid, CoreGraphics::BatchType, Animation::AnimJobEnqueueMode, CoreGraphics::Adapter, Http::HttpMethod, Input::Key, CoreGraphics::PixelFormat, Util::FourCC, Conditions::Operator< TYPE >, CoreAnimation::CurveType, CoreGraphics::IndexType, CoreGraphics::PrimitiveTopology,
CoreAnimation::InfinityType, Frame::LightingMode, CoreGraphics::ImageFileFormat, CoreGraphics::AntiAliasQuality, Lighting::LightType
- Front() : Util::SimpleTree<VALUETYPE>::Node, Util::List<TYPE>, Util::RingBuffer<TYPE>, Util::List<TYPE>, Util::Array<TYPE>
- frustum() : Math::frustum
- FuncPtr : Jobs::SerialJobFuncDesc, Jobs::TPJobFuncDesc
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_ a b c d e f g h i j k l m n o p q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- g -

- GameApplication() : App::GameApplication
- GameContentServer() : IO::GameContentServer
- GameContentServerBase() : Base::GameContentServerBase
- GamePadBase() : Base::GamePadBase
- GameServer() : Game::GameServer
- GameStateHandler() : BaseGameFeature::GameStateHandler
- GameThreadWaitForFrameSync() : Graphics::GraphicsInterface
- GatherSkinMesh() : Characters::CharacterServer
- GatherVisibleCharacter() : Characters::CharacterServer
- ge() : Math::float2
- gen() : Math::noise
- Generate() : OSX::OSXGUID, Win32::Win32Guid
- GenerateMipLevels() : Base::RenderTargetBase, Direct3D9::D3D9RenderTarget
- get() : WeakPtr< TYPE >, OSX::OSXThreadLocalPtr< TYPE >
- Get() : IO::BXmIReader, IO::XmlReader, Util::String
- get() : Ptr< TYPE >
- Get() : Models::VisResolveContainer< TYPE >
- get_cartesian() : Math::polar
- get_clipplanes() : Math::bbox
- get_position() : Math::matrix44
- get_unsafe() : Ptr< TYPE >, WeakPtr< TYPE >
- get_xaxis() : Math::matrix44
- get_yaxis() : Math::matrix44
- get_zaxis() : Math::matrix44
- GetAbsMousePos() : CoreGraphics::DisplayEvent, Input::InputEvent
- GetAbsoluteEndTime() : Animation::AnimJob
- GetAbsoluteStartTime() : Animation::AnimJob
- GetAbsoluteStopTime() : Animation::AnimJob
- GetAcceptedMessages() : Messaging::Port
- GetAccess() : Win360::D3D9StreamMeshLoader, Base::ResourceBase, CoreGraphics::MemoryMeshLoader, Win360::D3D9StreamMeshLoader, Base::ResourceBase
- GetAccessMode() : IO::Stream
- GetAccessPattern() : IO::Stream
- GetAction() : Script::ActionReader
- GetActionBock() : Script::DialogTake
- GetActionList() : Actions::SequenceAction, Actions::ActionList, Script::DialogTake
- GetActionRef() : Script::DialogTake
- GetActionString() : Script::ActionReader
- GetActiveShaderInstance() : Base::ShaderServerBase
- GetActiveVariation() : Base::ShaderInstanceBase, Characters::CharacterVariationSet, Base::ShaderInstanceBase
- GetAdapter() : Graphics::DisplaySettings, Base::DisplayDeviceBase
- GetAdapterInfo() : Direct3D9::D3D9DisplayDevice, Graphics::Display, Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice, Base::DisplayDeviceBase
- GetAddress() : Net::StdTcpServer, Win360::Win360Socket, Net::StdTcpServer
- GetAlignedBlockSize() : OSX::OSXMemoryPool, Win360::Win360MemoryPool
- GetAllAnimJobs() : Animation::AnimSequencer
- GetAllAssigns() : IO::AssignRegistry
- GetAllAllocatedMemory() : Resources::PoolResourceMapper
- GetAllRegisteredUriSchemes() : IO::SchemeRegistry
- GetAllShaderInstances() : Base::ShaderBase
- GetAllShaders() : Base::ShaderServerBase
- GetAlpha() :  CoreGraphics::MousePointer
- GetAmbientLightColor() :  Graphics::GlobalLightEntity, Lighting::InternalGlobalLightEntity
- GetAnimationResourceId() :  Characters::CharacterNode
- GetAnimDrivenMotionJointName() :  Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- GetAnimDrivenMotionVector() :  Graphics::ModelEntity, Characters::CharacterAnimationController, InternalGraphics::InternalModelEntity
- GetAnimEvent() :  Animation::AnimEventInfo
- GetAnimEvents() :  Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- GetAnimEventsByCategory() :  BaseGameFeature::AnimEventManager
- GetAnimJobName() :  Animation::AnimEventInfo
- GetAnimJobsByName() :  Animation::AnimSequencer
- GetAnimJobsByTrackIndex() :  Animation::AnimSequencer
- GetAnimResource() :  Animation::AnimSequencer
- GetAntiAliasQuality() :  Base::RenderTargetBase, Graphics::DisplaySettings, Base::DisplayDeviceBase, Base::RenderTargetBase, Base::DisplayDeviceBase
- GetAppDataDirectory() :  Win360::Win360FSWrapper, OSX::OSXFSWrapper
- GetAppID() :  App::Application
- GetAppName() :  Interface::InterfaceHandlerBase, Interface::InterfaceBase, Core::CoreServer, Interface::InterfaceHandlerBase, Interface::InterfaceBase, Interface::InterfaceHandlerBase, Interface::InterfaceBase
- GetAttrTable() :  Game::Entity
- GetAttrTableRowIndex() :  Game::Entity
- GetAssign() :  IO::AssignRegistry
- GetAttr() :  Game::Entity
- GetAttrName() :  IO::BXmLoaderUtil
- GetAttrs() :  IO::BXmReader, IO::XmlReader
- GetAttrTable() :  Game::Entity
- GetAttrTableRowIndex() :  Game::Entity
- `GetAttrValue()`: IO::BXmlLoaderUtil
- `GetAvailableDisplayModes()`: Direct3D9::D3D9DisplayDevice, Graphics::Display, Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice, Base::DisplayDeviceBase
- `GetAxisValue()`: Base::GamePadBase
- `GetBackLightColor()`: Graphics::GlobalLightEntity, Lighting::InternalGlobalLightEntity
- `GetBackLightOffset()`: Graphics::GlobalLightEntity, Lighting::InternalGlobalLightEntity
- `GetBaseIndex()`: CoreGraphics::PrimitiveGroup
- `GetBaseTime()`: Animation::AnimJob
- `GetBaseVertex()`: CoreGraphics::PrimitiveGroup
- `GetBatchByIndex()`: Frame::FramePassBase
- `GetBinDirectory()`: Win360::Win360FSWrapper, OSX::OSXFSWrapper
- `GetBlendWeight()`: Animation::AnimJob
- `GetBlob()`: BaseGameFeature::GlobalAttrsManager, Game::Entity, Util::Variant
- `GetBlobArray()`: Util::Variant
- `GetBlocking()`: Win360::Win360Socket
- `GetBlockSize()`: OSX::OSXMemoryPool, Win360::Win360MemoryPool
- `GetBool()`: BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity, Script::ActionReader, IO::BXmlReader, IO::XmlReader, Util::CommandLineArgs, Util::Variant, Particles::EmitterAttrs
- `GetBoolArray()`: Util::Variant
- `GetBoolAtIndex()`: Util::CommandLineArgs
- `GetBoolFlag()`: Util::CommandLineArgs
- `GetBoundingBox()`: CoreGraphics::PrimitiveGroup, Util::QuadTree< TYPE >::Node, Models::ModelNode, Models::Model, Models::ModelNode, Visibility::ObserverContext, Visibility::VisibilityContext
- `GetBroadcast()`: Win360::Win360Socket
- `GetBuffer()`: Util::RingBuffer< TYPE >
- `GetBufferSize()`: Jobs::JobDataDesc, Jobs::JobUniformDesc
- `GetByteSize()`: CoreAnimation::AnimKeyBuffer
- `GetCameraEntity()`: Graphics::View
**InternalGraphics::InternalView**, **Lighting::PSSMUtil**

- `GetCameraFocusEntity()`: **BaseGameFeature::FocusManager**
- `GetCameraSettings()`: **Graphics::CameraEntity**, **InternalGraphics::InternalCameraEntity**
- `GetCameraTransform()`: **RenderUtil::MayaCameraUtil**
- `GetCastShadows()`: **Graphics::AbstractLightEntity**, **Lighting::InternalAbstractLightEntity**
- `GetCastShadowsThisFrame()`: **Lighting::InternalAbstractLightEntity**
- `GetCategory()`: **Game::Entity**, **Characters::CharacterSkin**, **CoreAnimation::AnimEvent**
- `GetCategoryByIndex()`: **BaseGameFeature::CategoryManager**
- `GetCategoryByName()`: **BaseGameFeature::CategoryManager**
- `GetCategoryName()`: **Animation::AnimEventHandlerBase**
- `GetChar()`: **CoreGraphics::DisplayEvent**, **Input::InputEvent**
- `GetCharacter()`: **InternalGraphics::InternalModelEntity**, **Characters::CharacterNode**
- `GetCharacterInstance()`: **InternalGraphics::InternalModelEntity**, **Characters::CharacterNodeInstance**
- `GetCharInput()`: **Base::KeyboardBase**
- `GetCharJointInfos()`: **InternalGraphics::InternalModelEntity**
- `GetChildAt()`: **Util::QuadTree< TYPE >::Node**
- `GetChildren()`: **Models::ModelNode**, **Models::ModelNodeInstance**, **Models::ModelNode**, **Models::ModelNodeInstance**, **Models::ModelNodeInstance**, **Models::ModelNode**
- `GetChildTakes()`: **Script::DialogTake**
- `GetClass()`: **Script::ActionReader**
- `GetClassFourCC()`: **Core::RefCounted**
- `GetClassName()`: **Core::RefCounted**
- `GetClassRtti()`: **Core::Factory**
- `GetClearColor()`: **Base::RenderTargetBase**, **Frame::FramePassBase**, **Base::RenderTargetBase**
- `GetClearDepth()`: **Base::RenderTargetBase**, **Frame::FramePassBase**, **Base::MultipleRenderTargetBase**, **Base::RenderTargetBase**, **Base::MultipleRenderTargetBase**
- `GetClearFlags()`: **Base::RenderTargetBase**, **Frame::FramePassBase**, **Base::MultipleRenderTargetBase**, **Base::RenderTargetBase**, **Base::MultipleRenderTargetBase**
Frame::FramePassBase, Base::RenderTargetBase
- GetClearStencil(): Base::RenderTargetBase,
  Frame::FramePassBase, Base::RenderTargetBase,
  Frame::FramePassBase, Base::MultipleRenderTargetBase,
  Base::RenderTargetBase, Base::MultipleRenderTargetBase
- GetClientAddress(): Net::StdTcpClientConnection
- GetClientConnectionClass(): Net::StdTcpServer
- GetClientCount(): Resources::ManagedResource
- GetClipByIndex(): CoreAnimation::AnimResource
- GetClipByName(): CoreAnimation::AnimResource
- GetClipDuration(): CoreAnimation::AnimClip
- GetClipIndexByName(): CoreAnimation::AnimResource
- GetClipName(): Animation::PlayClipJob
- GetClipStatus(): Graphics::GraphicsEntity
- GetCloseConditions(): Script::Task
- GetCloseText(): Script::Task
- GetCmdLineArgs(): Game::FeatureUnit, App::Application,
  Game::FeatureUnit, App::Application, Game::FeatureUnit
- GetCmdName(): Util::CommandLineArgs
- GetCode(): Jobs::TPJobCommand
- GetColor(): CoreGraphics::RenderShape,
  CoreGraphics::TextElement, Graphics::AbstractLightEntity,
  Lighting::InternalAbstractLightEntity
- GetColorBufferFormat(): Base::RenderTargetBase
- GetColumnIndexByName(): Util::SparseTable< TYPE >
- GetCompanyName(): Interface::InterfaceHandlerBase,
  Interface::InterfaceBase, Core::CoreServer,
  App::Application, Interface::InterfaceHandlerBase,
  Interface::InterfaceBase, Interface::InterfaceHandlerBase,
  App::Application, Interface::InterfaceBase,
  Interface::InterfaceHandlerBase, Interface::InterfaceBase,
  Interface::InterfaceHandlerBase
- GetComponentAt(): Base::VertexLayoutBase
- GetCondition(): Script::ActionReader,
  Actions::IfThenElseAction, Conditions::Not,
  Script::DialogTake
- GetConditions(): Conditions::And, Conditions::Or
- GetContent(): IO::XmlReader
- GetContentLength(): Http::HttpResponseReader
- GetContentType(): Http::HttpResponseReader
- GetCoreId(): OSX::OSXThread, Win360::Win360Thread
- GetCornerIndexBuffer(): Direct3D9::D3D9ParticleRenderer
- GetCornerVertexBuffer(): Direct3D9::D3D9ParticleRenderer
- GetCount(): Net::DebugPacket
- GetCPUResolveTexture(): Base::RenderTargetBase
- GetCpuType(): Base::SystemInfoBase
- GetCurParticleVertexIndex(): Direct3D9::D3D9ParticleRenderer
- GetCurrentAdapterDisplayMode(): 
  Direct3D9::D3D9DisplayDevice, Base::DisplayDeviceBase, 
  Graphics::Display, Direct3D9::D3D9DisplayDevice, 
  Base::DisplayDeviceBase
- GetCurrentUserLightEntity(): 
  InternalGraphics::InternalGraphicsServer
- GetCurrentNodeIndex(): IO::BXmLoaderUtil
- GetCurrentNodeLineNumber(): IO::XmlReader
- GetCurrentNodeName(): IO::BXmLoaderUtil, 
  IO::BXmReader, IO::XmlReader
- GetCurrentNodePath(): IO::BXmReader, IO::XmlReader
- GetCurrentState(): FSM::StateMachine, 
  App::GameApplication
- GetCurrentStateHandler(): App::GameApplication
- GetCurrentTake(): Script::Dialog
- GetCurVertexPtr(): Direct3D9::D3D9ParticleRenderer
- GetCurveType(): CoreAnimation::AnimCurve
- GetD3D9BaseTexture(): Direct3D9::D3D9Texture
- GetD3D9CubeTexture(): Direct3D9::D3D9Texture
- GetD3D9Effect(): Direct3D9::D3D9ShaderVariation, 
  Direct3D9::D3D9Shader, Direct3D9::D3D9ShaderInstance, 
  Direct3D9::D3D9ShaderVariation, Direct3D9::D3D9Shader, 
  Direct3D9::D3D9ShaderInstance
- GetD3D9EffectPool(): Direct3D9::D3D9ShaderServer
- GetD3D9IndexBuffer(): Win360::D3D9IndexBuffer
- GetD3D9Technique(): Direct3D9::D3D9ShaderVariation
- GetD3D9Texture(): Direct3D9::D3D9Texture
- GetD3D9VertexBuffer(): Win360::D3D9VertexBuffer
- GetD3D9VertexDeclaration(): Win360::D3D9VertexLayout
- GetD3D9VolumeTexture(): Direct3D9::D3D9Texture
- GetDatabasePath() : BaseGameFeature::UserProfile
- GetDay() : Base::CalendarTimeBase
- GetDebugColor() : FSM::StateMachine
- GetDebugCounterByName() : Debug::DebugServer
- GetDebugCounters() : Debug::DebugServer
- GetDebugInfo() : Util::GlobalStringAtomTable
- GetDebugTextEnabled() : BaseGameFeature::LoaderServer
- GetDebugTimerByName() : Debug::DebugServer
- GetDebugTimers() : Debug::DebugServer
- GetDebugTxt() : Actions::Action, FSM::StateMachine, Actions::Action
- GetDefaultGamePad() : Base::InputServerBase
- GetDefaultKeyboard() : Base::InputServerBase
- GetDefaultMouse() : Base::InputServerBase
- GetDefaultRenderTarget() : Base::RenderTargetBase
- GetDefaultSamplesArray() : Characters::CharacterSkeleton
- GetDependencies() : InternalGraphics::InternalView
- GetDepth() : Base::TextureBase
- GetDepthResolveTexture() : Base::RenderTargetBase
- GetDesc() : Http::HttpRequestHandler
- GetDescription() : Script::InfoLog, CoreGraphics::AdapterInfo
- GetDeviceId() : CoreGraphics::AdapterInfo
- GetDeviceIndex() : Input::InputEvent
- GetDeviceName() : CoreGraphics::AdapterInfo
- GetDialog() : Script::DialogManager
- GetDialogId() : Script::DialogManager
- GetDialogLockState() : Script::DialogManager
- GetDialogTakeState() : Script::DialogManager
- GetDialogText() : Script::DialogManager
- GetDialogTexts() : Script::Dialog
- GetDirect3D() : Direct3D9::D3D9RenderDevice
- GetDirect3DDevice() : Direct3D9::D3D9RenderDevice
- GetDirEntries() : IO::ZipDirEntry
- GetDisplayMode() : Base::DisplayDeviceBase
- GetDriverName() : CoreGraphics::AdapterInfo
- GetDriverVersionHighPart() : CoreGraphics::AdapterInfo
- GetDriverVersionLowPart() : CoreGraphics::AdapterInfo
- GetDuration() : Animation::AnimJob
- GetElement() : Util::QuadTree< TYPE >::Node, IO::ExcelXmlReader
- GetElements() : Util::SparseTable< TYPE >
- GetElseBlock() : Actions::IfThenElseAction
- GetEmitterAttrs() : Particles::ParticleSystemNode, Particles::ParticleSystem
- GetEmitterMesh() : Particles::ParticleSystem
- GetEmitterMeshResourceId() : Particles::ParticleSystemNode
- GetEmitterPoint() : Particles::EmitterMesh
- GetEmote() : Script::DialogTake
- GetEnqueueMode() : Animation::AnimJob
- GetEntities() : BaseGameFeature::EntityManager, InternalGraphics::InternalGraphicsServer, InternalGraphics::InternalStage
- GetEntitiesByAttr() : BaseGameFeature::EntityManager
- GetEntitiesByAttrs() : BaseGameFeature::EntityManager
- GetEntitiesByType() : InternalGraphics::InternalStage
- GetEntitiesInActivityBubble() :
  BaseGameFeature::EntityManager
- GetEntitiesInBox() : BaseGameFeature::EnvQueryManager
- GetEntitiesInSphere() : BaseGameFeature::EnvQueryManager
- GetEntity() : Game::Property, Script::ActionReader, Actions::Action, Conditions::Condition, FSM::StateMachine, Actions::Action, Game::Property, Conditions::Condition, Game::Property
- GetEntityByAttr() : BaseGameFeature::EntityManager
- GetEntityByAttrs() : BaseGameFeature::EntityManager
- GetEntityById() : InternalGraphics::InternalGraphicsServer
- GetEntityById() : BaseGameFeature::EntityManager
- GetEntityId() : Animation::AnimEventInfo
- GetEntityUnderMouse() :
  BaseGameFeature::EnvQueryManager
- GetEntry() : Resources::ResourceDictionary
- GetEnvelope() : Particles::EmitterAttrs
- GetEnvelopeSampleBuffer() : Particles::ParticleSystem
- GetErrorCode() : Win360::Win360Socket
- GetErrorString() : Win360::Win360Socket
- GetEventByIndex() : CoreAnimation::AnimClip
- GetEventByName() : CoreAnimation::AnimClip
- GetEventCode() : CoreGraphics::DisplayEvent, CoreGraphics::RenderEvent
- GetEventsInRange() : CoreAnimation::AnimClip
- GetExclusiveTag() : Animation::AnimJob
- GetFactor() : BaseGameFeature::TimeSource
- GetFadeInTime() : Animation::AnimJob
- GetFadeOutTime() : Animation::AnimJob
- GetFailConditions() : Script::Task
- GetFailedText() : Script::Task
- GetFarHeight() : Shared::CameraSettings
- GetFarWidth() : Shared::CameraSettings
- GetFeatureBits() : Base::ShaderServerBase
- GetFeatureMask() : Base::ShaderVariationBase
- GetFileEntries() : IO::ZipDirEntry
- GetFileExtension() : Util::String
- GetFileName() : StateObjectFeature::StateProperty
- GetFileSize() : Win360::Win360FSWrapper, IO::ZipFileEntry, OSX::OSXFSWrapper
- GetFileWriteTime() : Win360::Win360FSWrapper, IO::IoServer, OSX::OSXFSWrapper
- GetFirstKeyIndex() : CoreAnimation::AnimCurve
- GetFirstSlice() : Jobs::TPJobCommand
- GetFittingPoolIndex() : Resources::PoolResourceMapper
- GetFloat() : BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity, Script::ActionReader, Util::CommandLineArgs, IO::XmlReader, Particles::EmitterAttrs, IO::BXmlReader, Util::Variant
- GetFloat2() : IO::BXmlReader, IO::XmlReader
- GetFloat4() : BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity, Script::ActionReader, Util::CommandLineArgs, IO::XmlReader, IO::BXmlReader, Util::Variant
- GetFloat4Array() : Util::Variant
- GetFloat4AtIndex() : Util::CommandLineArgs
- GetFloatArray() : Util::Variant
- GetFloatAtIndex(): Util::CommandLineArgs
- GetFocalLength(): Base::TransformDeviceBase, Shared::CameraSettings
- GetFocusEntity(): BaseGameFeature::FocusManager
- GetFormat(): Base::StreamTextureSaverBase
- GetFourCC(): Core::Rtti
- GetFov(): Shared::CameraSettings
- GetFragmentJointPalette(): Characters::CharacterSkinNode
- GetFragmentPrimGroupIndex(): Characters::CharacterSkinNode
- GetFrameActions(): FSM::State
- GetFrameCount(): FrameSync::FrameSyncHandlerThread, FrameSync::FrameSyncTimer, Graphics::GraphicsServer
- GetFrameId(): BaseGameFeature::TimeSource
- GetFramePassBaseByIndex(): Frame::FrameShader
- GetFramePassBaseByName(): Frame::FrameShader
- GetFrameShader(): InternalGraphics::InternalView
- GetFrameShaderName(): Graphics::View
- GetFrameTicks(): FrameSync::FrameSyncTimer
- GetFrameTime(): FrameSync::FrameSyncTimer, App::RenderApplication, BaseGameFeature::TimeSource, App::RenderApplication, BaseGameFeature::TimeSource
- GetFrequency(): GraphicsFeature::LightFlickerUtil
- GetFromByteOrder(): System::ByteOrder
- GetFuncDesc(): Base::JobBase
- GetFunctionPointer(): Jobs::SerialJobFuncDesc, Jobs::TPJobFuncDesc
- GetGfxEntity(): Visibility::VisibilityContext
- GetGlobalLightShadowBufferTexture(): Lighting::SM30ShadowServer
- GetGlobalMatrix(): Shared::CharJointInfo
- GetGlobalStringBuffer(): Util::GlobalStringAtomTable
- GetGraphicsEntities(): BaseGameFeature::EnvEntityManager, Visibility::VisibilityContainer
- GetGraphicsResource(): GraphicsFeature::GraphicsProperty
- GetGroup(): Script::Dialog, Script::DialogDesc
- GetGuid(): BaseGameFeature::GlobalAttrsManager
Game::Entity, Script::Dialog, Script::DialogDesc, Script::DialogTake, Script::Task, Util::Variant, CoreGraphics::AdapterInfo
- GetGuidArray() : Util::Variant
- GetHandlerAtIndex() : Messaging::Port
- GetHandlers() : IO::Console
- GetHandlerThread() : Messaging::AsyncPort
- GetHeight() : Base::RenderTargetBase, Base::TextureBase, CoreGraphics::DisplayMode, Base::RenderTargetBase
- GetHighFrequencyVibrator() : Base::GamePadBase
- GetHistory() : Debug::DebugTimer, IO::HistoryConsoleHandler, Debug::DebugCounter, IO::HistoryConsoleHandler
- GetHistorySize() : IO::HistoryConsoleHandler
- GetHomeDirectory() : Win360::Win360FSWrapper, OSX::OSXFSWrapper
- GetHostAddr() : Win360::Win360IpAddress
- GetHostAndLocalPath() : IO::URI
- GetHostName() : Win360::Win360IpAddress
- GetHotspot() : CoreGraphics::MousePointer
- GetHour() : Base::CalendarTimeBase
- GetHttpMethod() : Http::HttpRequestReader
- GetHwnd() : Win32::Win32DisplayDevice
- GetIconName() : Base::DisplayDeviceBase, Graphics::DisplaySettings, Base::DisplayDeviceBase
- GetId() : Script::Dialog, Script::DialogDesc, Script::DialogTake, Script::Task, InternalGraphics::InternalGraphicsEntity
- GetIndexBuffer() : Base::RenderDeviceBase, Base::MeshBase, Base::RenderDeviceBase, Base::MeshBase
- GetIndexData() : CoreGraphics::RenderShape
- GetIndexType() : Base::IndexBufferBase, CoreGraphics::RenderShape, Base::IndexBufferBase
- GetInfo() : Script::InfoLog
- GetInput() : Win32::Win32ConsoleHandler, IO::ConsoleHandler, IO::Console, IO::ConsoleHandler, OSX::OSXConsoleHandler
- GetInputDesc() : Base::JobBase
- GetInputFocusEntity() : BaseGameFeature::FocusManager
- GetInstanceDataset(): BaseGameFeature::CategoryManager::Category
- GetInstanceEntity(): BaseGameFeature::CategoryManager
- GetInstances(): Models::Model
- GetInstancesByAttr(): BaseGameFeature::CategoryManager
- GetInstancesByAttrs(): BaseGameFeature::CategoryManager
- GetInstanceSize(): Core::Rtti
- GetInstanceTable(): BaseGameFeature::CategoryManager
- GetInstanceTableName(): BaseGameFeature::CategoryManager::Category
- GetInt(): BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity, Script::ActionReader, Util::CommandLineArgs, IO::XmlReader, IO::BXmlReader, Util::Variant, Particles::EmitterAttrs
- GetIntArray(): Util::Variant
- GetIntAtIndex(): Util::CommandLineArgs
- GetIntensityAmplitude(): GraphicsFeature::LightFlickerUtil
- GetInvLightProjTransform(): Lighting::InternalAbstractLightEntity
- GetInvModelTransform(): Base::TransformDeviceBase
- GetInvModelViewTransform(): Base::TransformDeviceBase
- GetInvPoseMatrix(): Characters::CharacterJoint
- GetInvPoseMatrixArray(): Characters::CharacterSkeleton
- GetInvProjTransform(): Base::TransformDeviceBase, Shared::CameraSettings
- GetInvTransform(): Lighting::InternalAbstractLightEntity
- GetInvViewTransform(): Base::TransformDeviceBase
- GetJob(): Jobs::TPJobSlice
- GetJobPort(): Characters::CharacterInstance
- GetJoint(): Characters::CharacterSkeleton
- GetJointIndexByName(): Characters::CharacterSkeleton
- GetJointMatrix(): Characters::CharacterSkeletonInstance
- GetJointName(): Shared::CharJointInfo
- GetJointTextureRowIndex(): Characters::CharacterInstance
- GetKeepAlive(): Win360::Win360Socket
- GetKey(): CoreGraphics::DisplayEvent, Input::InputEvent
- GetKeyboardCaptureHandler(): Base::InputServerBase
- GetKeyBuffer(): CoreAnimation::AnimResource
- GetKeyBufferPointer() : CoreAnimation::AnimKeyBuffer
- GetKeyDuration() : CoreAnimation::AnimClip
- GetKeySliceByteSize() : CoreAnimation::AnimClip
- GetKeySliceFirstKeyIndex() : CoreAnimation::AnimClip
- GetKeyStride() : CoreAnimation::AnimClip
- GetLastFrameId() : Resources::ManagedResource
- GetLevelName() : BaseGameFeature::GameStateHandler
- GetLightDir() : Lighting::PSSMUtil
- GetLightDirection() : Lighting::InternalGlobalLightEntity
- GetLightEntity() : GraphicsFeature::LightFlickerUtil
- GetLightingMode() : Frame::FrameBatch
- GetLightType() : Graphics::AbstractLightEntity, Lighting::InternalAbstractLightEntity
- GetLinks() : InternalGraphics::InternalGraphicsEntity
- GetLoadedResource() : Resources::ManagedResource
- GetLoader() : Resources::Resource
- GetLocalBoundingBox() : Graphics::GraphicsEntity, InternalGraphics::InternalGraphicsEntity
- GetLocalLightShadowBufferTexture() : Lighting::SM30ShadowServer
- GetLocalMatrices() :
  PhysicsFeature::EnvironmentCollideProperty
- GetLocalMatrix() : Shared::CharJointInfo
- GetLocalTime() : Base::CalendarTimeBase, Win360::Win360CalendarTime
- GetLocalTransform() : Models::TransformNodeInstance
- GetLockedToViewer() : Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode
- GetLogLevel() : Script::InfoLog
- GetLoopCount() : Animation::PlayClipJob
- GetLowFrequencyVibrator() : Base::GamePadBase
- GetMagic() : Net::DebugPacket
- GetMainRenderTarget() : Frame::FrameShader
- GetManagedAnimResource() : Characters::CharacterNode
- GetManagedMesh() : Models::ShapeNode
- GetManagedResource() : Resources::LoadingResource
- `GetMapperByResourceType` : `Resources::ResourceManager`
- `getmatrix` : `Math::transform44`
- `GetMatrix44()` : `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`, `Util::CommandLineArgs`, `IO::XmlReader`, `IO::BXmlReader`, `Util::Variant`
- `GetMatrix44Array()` : `Util::Variant`
- `GetMatrix44AtIndex()` : `Util::CommandLineArgs`
- `GetMaxDistance()` : `Models::TransformNode`
- `GetMaxMsgSize()` : `Win360::Win360Socket`
- `GetMaxNumLocalPlayers` : `Base::InputServerBase`
- `GetMaxNumParticles()` : `Particles::ParticleSystem`
- `GetMaxNumPlayers()` : `Base::GamePadBase`
- `GetMaxProgressValue()` : `BaseGameFeature::LoaderServer`
- `GetMaxScreenSpaceSize()` : `Resources::ManagedResource`
- `GetMaxTriggerDistance()` : `BaseGameFeature::EntityManager`
- `GetMaxValue()` : `Particles::EnvelopeCurve`
- `GetMediaType()` : `IO::Stream`
- `GetMemoryPool()` : `Memory::PoolArrayAllocator`
- `GetMemorySize()` : `Base::RenderTargetBase`
- `GetMesh()` : `Resources::ManagedMesh`
- `GetMeshResourceId()` : `Models::ShapeNode`
- `GetMessages()` : `Messaging::BatchMessage`
- `GetMethod()` : `Http::HttpRequest`
- `GetMilliSecond()` : `Base::CalendarTimeBase`
- `GetMinDistance()` : `Models::TransformNode`
- `GetMinute()` : `Base::CalendarTimeBase`
- `GetMipLevel()` : `Base::StreamTextureSaverBase`
- `GetModel()` : `Models::ModelNode`, `Models::ManagedModel`, `Models::ModelInstance`, `Models::ModelNode`
- `GetModelEntity()` : `Models::ModelInstance`
- `GetModelInstance()` : `Models::ModelNodeInstance`, `InternalGraphics::InternalModelEntity`, `Models::ModelNodeInstance`
- `GetModelNode()` : `Characters::CharacterSkin`, `Models::ModelNodeInstance`
- `GetModelNodeInstanceIndex()` : `Models::ModelNodeInstance`
- `GetModelResourceMapper()` : `Models::ModelServer`
- `GetModelResourceState()` : `InternalGraphics::InternalModelEntity`
- `GetModelTransform()` : `Base::TransformDeviceBase`, `CoreGraphics::RenderShape`, `Models::TransformNodeInstance`
- `GetModelViewProjTransform()` : `Base::TransformDeviceBase`
- `GetModelViewTransform()` : `Base::TransformDeviceBase`
- `GetMonth()` : `Base::CalendarTimeBase`
- `GetMouseButton()` : `Input::InputEvent`, `CoreGraphics::DisplayEvent`
- `GetMouseCaptureHandler()` : `Base::InputServerBase`
- `GetMouseMovement()` : `Win32::Win32InputServer`
- `GetMousePos3d()` : `BaseGameFeature::EnvQueryManager`
- `GetMovement()` : `Win32::Win32Mouse`, `Base::MouseBase`, `Win32::Win32Mouse`
- `GetMRTIndex()` : `Base::RenderTargetBase`
- `GetMultipleRenderTarget()` : `Frame::FramePassBase`, `InternalGraphics::InternalView`, `Frame::FramePassBase`
- `GetMultipleRenderTargetByIndex()` : `Frame::FrameShader`
- `GetMultipleRenderTargetByName()` : `Frame::FrameShader`
- `GetMyThreadId()` : `OSX::OSXThread`, `Win360::Win360Thread`
- `GetMyThreadName()` : `Win360::Win360Thread`
- `GetName()` : `BaseGameFeature::UserProfile`, `Base::ShaderVariableBase`, `IO::Assign`, `Base::ShaderVariableBase`, `BaseGameFeature::CategoryManager::Category`, `Base::ShaderVariationBase`, `Debug::DebugTimer`, `Base::ShaderVariableBase`, `Base::ShaderVariationBase`, `Http::HttpRequestHandler`, `OSX::OSXThread`, `FSM::State`, `FSM::StateMachine`, `Http::HttpRequestHandler`, `Win360::Win360Thread`, `Http::HttpRequestHandler`, `Models::ModelNodeInstance`, `Http::HttpRequestHandler`, `Frame::FramePassBase`, `Animation::AnimJob`, `Core::Rtti`, `Win360::Win360Thread`, `Models::ModelNodeInstance`, `Frame::FramePassBase`, `CoreAnimation::AnimEvent`, `Frame::FrameShader`, `IO::ZipDirEntry`, `Win360::Win360Heap`, `Http::HttpRequestHandler`, `IO::ZipFileEntry`, `Win360::Win360Heap`, `Http::HttpRequestHandler`, `IO::ZipFileEntry`, `Characters::CharacterSkin`, `Models::ModelNodeInstance`, `Http::HttpRequestHandler`, `Win360::Win360Thread`, `Graphics::View`, `Models::ModelNode`, `Win360::Win360Thread`, `Animation::AnimJob`,
Models::ModelNode, Debug::DebugCounter, HTTP::HttpRequestHandler, Models::ModelNode, Characters::CharacterJoint, Models::ModelNodeInstance, Win360::Win360Thread, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Characters::CharacterSkinList, Win360::Win360Thread, InternalGraphics::InternalView, Models::ModelNode, Win360::Win360Thread, Models::ModelNodeInstance, Models::ModelNode, CoreAnimation::AnimClip, InternalGraphics::InternalStage, OSX::OSXHeap

- GetNearHeight(): Shared::CameraSettings
- GetNearWidth(): Shared::CameraSettings
- GetNewSlot(): Resources::PoolLoadingResource
- GetNextThreadIndex(): Jobs::TPJobThreadPool
- GetNodeFilter(): Frame::FrameBatch
- GetNodeInstances(): Models::ModelInstance
- GetNoDelay(): Win360::Win360Socket
- GetNodeName(): IO::BXmlLoaderUtil
- GetNodes(): Models::Model
- GetNormMousePos(): CoreGraphics::DisplayEvent, Input::InputEvent
- GetNumArgs(): Util::CommandLineArgs
- GetNumAttrs(): IO::BXmlLoaderUtil
- GetNumBatches(): Frame::FramePassBase
- GetNumBlocks(): Win360::Win360MemoryPool, OSX::OSXMemoryPool
- GetNumBuffers(): Jobs::JobDataDesc, Jobs::JobUniformDesc
- GetNumCategories(): BaseGameFeature::CategoryManager
- GetNumChildTakes(): Script::DialogTake
- GetNumChunks(): Util::StringBuffer
- GetNumClips(): CoreAnimation::AnimResource
- GetNumColumns(): Util::SparseTable< TYPE >, IO::ExcelXmlReader
- GetNumComponents(): Base::VertexLayoutBase
- GetNumCpuCores(): Base::SystemInfoBase
- GetNumCurves(): CoreAnimation::AnimClip
- GetNumEvents(): CoreAnimation::AnimClip
- GetNumFragments(): Characters::CharacterSkinNode
- GetNumFramePassBases(): Frame::FrameShader
- GetNumHandlers(): Messaging::Port
- GetNumIndices(): CoreGraphics::PrimitiveGroup, Base::IndexBufferBase
- GetNumInstances(): BaseGameFeature::CategoryManager
- GetNumJoints(): Characters::CharacterSkeleton, Characters::CharacterSkeletonInstance
- GetNumKeys(): CoreAnimation::AnimClip, CoreAnimation::AnimKeyBuffer
- GetNumMipLevels(): Base::TextureBase
- GetNumMultipleRenderTargets(): Frame::FrameShader
- GetNumPasses(): Base::ShaderVariationBase
- GetNumPendingResources(): Resources::ResourceMapper, Resources::SimpleResourceMapper
- GetNumPools(): Resources::PoolResourceMapper
- GetNumPrimitiveGroups(): Base::MeshBase
- GetNumPrimitives(): CoreGraphics::PrimitiveGroup, CoreGraphics::RenderShape
- GetNumRendertargets(): Base::MultipleRenderTargetBase
- GetNumRenderTargets(): Frame::FrameShader
- GetNumRendertargets(): Base::MultipleRenderTargetBase
- GetNumRows(): Util::SparseTable< TYPE >, Resources::D3D9TextureStreamer, IO::ExcelXmlReader
- GetNumSamples(): CoreAnimation::AnimSampleBuffer, Particles::EnvelopeSampleBuffer
- GetNumSharedVariables(): Base::ShaderServerBase
- GetNumSkinLists(): Characters::CharacterSkinLibrary
- GetNumSkins(): Characters::CharacterSkinLibrary, Characters::CharacterSkinSet
- GetNumSlices(): Jobs::TPJobCommand
- GetNumStates(): App::GameApplication
- GetNumSubTasks(): Script::Task
- GetNumTables(): IO::ExcelXmlReader
- GetNumTextures(): Frame::FrameShader
- GetNumTimeSources(): BaseGameFeature::TimeManager
- GetNumTransitions(): FSM::State
- GetNumUniqueElements(): Util::SparseTable< TYPE >
- GetNumVariables(): Base::ShaderInstanceBase, Frame::FramePassBase, Base::ShaderInstanceBase,
Frame::FrameBatch, Base::ShaderInstanceBase
- GetNumVariations(): Base::ShaderInstanceBase
- GetNumVertices(): Base::VertexBufferBase, CoreGraphics::PrimitiveGroup, CoreGraphics::RenderShape
- GetObject(): Util::Variant
- GetObjectRef(): Graphics::GraphicsEntity
- GetObserver(): Visibility::VisibilityQuery
- GetObserverBitMask(): Visibility::VisibilityBoxSystem, Visibility::VisibilityClusterSystem, Visibility::VisibilityQuadtree, Visibility::VisibilitySystemBase
- GetObserverEntity(): Visibility::ObserverContext
- GetObserverType(): Visibility::VisibilityQuery
- getoffset(): Math::transform44
- GetOffsetMatrix(): Models::TransformNodeInstance
- GetOldSlot(): Resources::PoolLoadingResource
- GetOpenConditions(): Script::Task
- GetOpenText(): Script::Task
- GetOptBool(): IO::XmlReader, IO::BXmlReader
- GetOptFloat(): IO::XmlReader, IO::BXmlReader
- GetOptFloat2(): IO::XmlReader, IO::BXmlReader
- GetOptFloat4(): IO::XmlReader, IO::BXmlReader
- GetOptInt(): IO::XmlReader, IO::BXmlReader
- GetOptMatrix44(): IO::BXmlReader, IO::XmlReader
- GetOptString(): IO::XmlReader, IO::BXmlReader
- GetOrientation(): CoreGraphics::MousePointer
- GetOriginalShader(): Base::ShaderInstanceBase
- GetOutputDesc(): Base::JobBase
- GetPageSize(): Base::SystemInfoBase
- GetParent(): Models::ModelNodeInstance, Core::Rtti, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode
- GetParentGuid(): Script::Task
- GetParentJoint(): Characters::CharacterJoint
- GetParentJointIndex(): Characters::CharacterJoint
- GetParentKey(): Script::Task
- GetParentNodeIndex(): IO::BXmlLoaderUtil
- GetParentWindow(): Base::DisplayDeviceBase,
Graphics::Display , Base::DisplayDeviceBase
- GetParticleSystem() : Particles::ParticleSystemNode
- GetParticleSystemInstance() :
  Particles::ParticleSystemNodeInstance
- GetParticleVertexBuffer() :
  Direct3D9::D3D9ParticleRenderer
- GetPassRenderTarget() :
  Base::RenderDeviceBase
- GetPath() :
  IO::Assign
- GetPayload() :
  Net::DebugPacket
- GetPayloadSize() :
  Net::DebugPacket
- GetPhysicsEntity() :
  PhysicsFeature::ActorPhysicsProperty ,
  PhysicsFeature::PhysicsProperty
- GetPixelFormat() :
  Base::TextureBase ,
  CoreGraphics::DisplayMode , Base::TextureBase
- GetPixelPosition() :
  Base::MouseBase
- GetPlaceholderResourceId() :
  Resources::ResourceMapper
- GetPlatform() :
  Base::SystemInfoBase
- GetPlayerIndex() :
  Base::GamePadBase
- GetPointer() :
  Jobs::JobUniformDesc , Jobs::JobDataDesc
- GetPoolForDebug() :
  Resources::PoolResourceMapper
- GetPoolSize() :
  Win360::Win360MemoryPool ,
  OSX::OSXMemoryPool
- GetPort() :
  Win360::Win360IpAddress , Net::DebugPacket ,
  Http::HttpServer , Net::DebugMessage
- GetPoseMatrix() :
  Characters::CharacterJoint
- GetPoseRotation() :
  Characters::CharacterJoint
- GetPoseScale() :
  Characters::CharacterJoint
- GetPoseTranslation() :
  Characters::CharacterJoint
- GetPosition() :
  IO::MemoryStream ,
  Models::TransformNodeInstance ,
  CoreGraphics::MouseListener , Models::TransformNode ,
  CoreGraphics::TextElement ,
  Models::TransformNodeInstance , Models::TransformNode ,
  Models::TransformNodeInstance , IO::FileStream ,
  Models::TransformNodeInstance , IO::FileStream
- getPosition() :
  Math::transform44
- GetPosition() :
  Models::TransformNodeInstance ,
  Models::TransformNode , Models::TransformNodeInstance ,
  Visibility::ObserverContext , IO::Stream
- GetPositionAmplitude() :
  GraphicsFeature::LightFlickerUtil
- `GetPostInfinityType()` : `CoreAnimation::AnimClip`
- `GetPreInfinityType()` : `CoreAnimation::AnimClip`
- `GetPresentParams()` : `Direct3D9::D3D9AnimClip`
- `GetPrimitiveGroup()` : `Base::RenderDeviceBase`, `Direct3D9::D3D9ParticleRenderer`, `Base::RenderDeviceBase`
- `GetPrimitiveGroupAtIndex()` : `Base::MeshBase`
- `GetPrimitiveGroupIndex()` : `Particles::ParticleSystemNode`, `Models::ShapeNode`
- `GetPrimitiveTopology()` : `CoreGraphics::PrimitiveGroup`
- `GetPriority()` : `Win360::Win360Thread`, `OSX::OSXThread`, `Resources::LoadingResource`, `Win360::Win360Thread`, `Resources::ManagedResource`, `Win360::Win360Thread`, `Resources::ManagedResource`, `Win360::Win360Thread`, `Resources::LoadingResource`
- `GetPrivateBuffer()` : `Base::JobBase`
- `GetPrivateBufferSize()` : `Base::JobBase`
- `GetProfileDirectory()` : `BaseGameFeature::UserProfile`
- `GetProfileRootDirectory()` : `BaseGameFeature::UserProfile`
- `GetProgramsDirectory()` : `Win360::Win360FSWrapper`, `OSX::OSXFSWrapper`
- `GetProgressResource()` : `BaseGameFeature::LoaderServer`
- `GetProgressText()` : `BaseGameFeature::LoaderServer`
- `GetProjectionMatrix()` : `Visibility::ObserverContext`
- `GetProjMapUvOffsetAndScale()` : `Graphics::AbstractLightEntity`, `Lighting::InternalAbstractLightEntity`
- `GetProjTransform()` : `Base::TransformDeviceBase`, `InternalGraphics::InternalCameraEntity`, `Base::TransformDeviceBase`, `Lighting::InternalAbstractLightEntity`, `Graphics::CameraEntity`, `Shared::CameraSettings`
- `GetPSSMSplitDistances()` : `Lighting::SM30ShadowServer`
- `GetPSSMSplitLightProjTransforms()` : `Lighting::SM30ShadowServer`
- `GetPtr()` : `Util::Blob`
- `GetRawBuffer()` : `Net::DebugPacket`
- `GetRawPointer()` : `IO::MemoryStream`
- `GetRecvBufSize()` : `Win360::Win360Socket`
- `GetRecvStream()`: `Net::MessageClient`, `Net::MessageClientConnection`, `Net::StdTcpClient`, `Net::StdTcpClientConnection`, `Net::StdTcpClient`, `Net::StdTcpClientConnection`
- `GetRefCount()`: `Core::RefCounted`
- `GetRenderCount()`: `Resources::ManagedResource`
- `GetRenderDebug()`: `InternalGraphics::InternalGraphicsServer`
- `GetRenderModules()`: `Graphics::GraphicsServer`
- `GetRenderTarget()`: `Frame::FramePassBase`, `InternalGraphics::InternalView`, `Base::MultipleRenderTargetBase`
- `GetRenderTargetByIndex()`: `Frame::FrameShader`
- `GetRenderTargetByName()`: `Frame::FrameShader`
- `GetRequestHandlers()`: `Http::HttpServer`
- `GetRequestURI()`: `Http::HttpRequestReader`
- `GetResolveDepthTextureResourceId()`: `Base::RenderTargetBase`
- `GetResolveRect()`: `Base::RenderTargetBase`
- `GetResolveTargetCpuAccess()`: `Base::RenderTargetBase`
- `GetResolveTexture()`: `Base::RenderTargetBase`
- `GetResolveTextureHeight()`: `Base::RenderTargetBase`
- `GetResolveTextureResourceId()`: `Base::RenderTargetBase`
- `GetResolveTextureWidth()`: `Base::RenderTargetBase`
- GetResourcesByType() : Resources::ResourceManager
- GetResourceState() : Models::TransformNode, Models::ShapeNode, Models::ModelNode, Characters::CharacterNode, Models::StateNode, Particles::ParticleSystemNode, Models::ShapeNode
- GetResourceStreamingLevelOfDetail() : Resources::ManagedResource
- GetResourceType() : Resources::ResourceMapper, Resources::SimpleResourceMapper, Resources::ManagedResource
- GetResponseAtIndex() : Script::Dialog
- GetResponseContentStream() : Http::HttpRequest
- GetResponsePassedStates() : Script::DialogManager, Script::Dialog
- GetResponses() : Script::DialogManager
- GetResponseSoundIDs() : Script::Dialog, Script::DialogManager
- GetResponseSpeakers() : Script::DialogManager, Script::Dialog
- GetResponseTexts() : Script::Dialog
- GetResult() : Animation::AnimSequencer, Conditions::Condition, Util::Crc, Conditions::Condition
- GetReturnCode() : App::Application
- GetReUseAddr() : Win360::Win360Socket
- GetReuseTexture() : Resources::D3D9TextureStreamer
- GetRevision() : CoreGraphics::AdapterInfo
- GetRootDirectory() : Core::CoreServer, Interface::InterfaceBase
- GetRootLocation() : Http::HttpRequestHandler
- GetRootNode() : Models::Model
- GetRootNodeInstance() : Models::ModelInstance
- GetRootNodeOffsetMatrix() : Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- GetRootNodePath() : Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- GetRotate() : Models::TransformNodeInstance
- getrotate() : Math::transform44
- GetRotate() : Models::TransformNodeInstance
- GetRotatePivot() : Models::TransformNode,
Models::TransformNodeInstance, Models::TransformNode
- getrotaterepivot() : Math::transform44
- GetRotatePivot() : Models::TransformNodeInstance,
  Models::TransformNode, Models::TransformNodeInstance,
  Models::TransformNode
- GetRotation() : Models::TransformNode
- getrow0() : Math::matrix44
- getrow1() : Math::matrix44
- getrow2() : Math::matrix44
- getrow3() : Math::matrix44
- GetRowIndexByName() : Util::SparseTable< TYPE >
- GetRTPlugins() : RenderModules::RTPluginRegistry
- GetSafeRLEBufferSize() : Util::RunLengthCodec
- GetSample() : Debug::DebugTimer, Debug::DebugCounter
- GetSampleBuffer() : Particles::EnvelopeSampleBuffer
- GetSampleCountsPointer() :
  CoreAnimation::AnimSampleBuffer
- GetSamplesPointer() : CoreAnimation::AnimSampleBuffer
- GetSaveActionsFlag() : Actions::SequenceAction
- GetSaveGame() : BaseGameFeature::GameStateHandler
- GetSaveGameDirectory() : BaseGameFeature::UserProfile
- GetSaveGamePath() : BaseGameFeature::UserProfile
- GetSaver() : Resources::Resource
- GetScale() : Models::TransformNode,
  Models::TransformNodeInstance
- getscale() : Math::transform44
- GetScale() : Models::TransformNodeInstance,
  Models::TransformNode, Models::TransformNodeInstance,
  Models::TransformNode
- GetScaledFrameTime() : FrameSync::FrameSyncTimer
- GetScaledTime() : FrameSync::FrameSyncTimer
- GetScalePivot() : Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode
- getscalepivot() : Math::transform44
- GetScalePivot() : Models::TransformNodeInstance,
  Models::TransformNode, Models::TransformNodeInstance
- GetScratchSize() : Jobs::JobUniformDesc
- GetScreenPosition() : Base::MouseBase
- GetScreenshotFilepath(): BaseGameFeature::LoaderServer
- GetSecond(): Base::CalendarTimeBase
- GetSemantic(): Base::ShaderVariableBase
- GetSendBufSize(): Win360::Win360Socket
- GetSendStream(): Net::StdTcpClient, Net::StdTcpClientConnection, Net::StdTcpClient, Net::MessageClient, Net::MessageClientConnection
- GetServerAddress(): Net::StdTcpClient
- GetSetupMode(): BaseGameFeature::GameStateHandler
- GetShader(): Models::StateNode, Frame::FramePassBase, Frame::FrameBatch, Models::StateNode, Frame::FramePassBase
- GetShaderFeatures(): Frame::FrameBatch
- GetShaderInstance(): Models::StateNode
- GetShaderParameter(): Models::StateNode
- GetShaderVariable(): Base::ShaderVariableInstanceBase
- GetShaderVariableInstance(): Models::StateNodeInstance
- GetShadowBufferUvOffsetAndScale(): Lighting::InternalAbstractLightEntity
- GetShadowIntensity(): Lighting::InternalAbstractLightEntity
- GetShadowInvLightProjTransform(): Lighting::InternalAbstractLightEntity
- GetShadowInvTransform(): Lighting::InternalAbstractLightEntity
- GetShadowProjTransform(): Lighting::InternalAbstractLightEntity
- GetShapes(): PhysicsFeature::EnvironmentCollideProperty
- GetShapeType(): CoreGraphics::RenderShape
- GetSharedVariableByIndex(): Base::ShaderServerBase
- GetSharedVariableBySemantic(): Base::ShaderServerBase
- GetShortResponseTexts(): Script::Dialog, Script::DialogManager
- GetShortText(): Script::DialogTake
-GetSize(): Resources::ResourceDictionary::Entry, CoreGraphics::MousePointer, IO::Stream, IO::ZipFileStream, IO::FileStream, IO::MemoryStream
- GetSkin(): Characters::CharacterSkinSet, Characters::CharacterSkinLibrary
- GetSkinByName(): Characters::CharacterSkinLibrary
- GetSkinIndexByName() : Characters::CharacterSkinLibrary
- GetSkinList() : Characters::CharacterSkinLibrary
- GetSkinListByName() : Characters::CharacterSkinLibrary
- GetSkinListIndexByName() : Characters::CharacterSkinLibrary
- GetSkinMatrixArray() : Characters::CharacterSkeletonInstance
- GetSkinningTechnique() : Characters::CharacterServer, Win32::Win32SkinnedMeshRenderer, Base::SkinnedMeshRendererBase, Win32::Win32SkinnedMeshRenderer
- GetSkinNodeInstanceByIndex() : Characters::CharacterSkinSet
- GetSkinNodeInstanceByName() : Characters::CharacterSkinSet
- GetSkins() : Characters::CharacterSkinList
- GetSkippedMips() : Base::TextureBase
- GetSliceIndex() : Jobs::TPJobSlice
- GetSliceSize() : Jobs::JobDataDesc
- GetSortingMode() : Frame::FrameBatch
- GetSound() : Script::DialogTake
- GetSource() : Script::InfoLog
- GetSpeaker() : Script::Dialog, Script::DialogTake, Script::DialogDesc, Script::DialogManager
- GetSplitDistances() : Lighting::PSSMUtil
- GetSplitLightProjTransform() : Lighting::PSSMUtil
- GetSplitLightProjTransforms() : Lighting::PSSMUtil
- GetSplitLightTransform() : Lighting::PSSMUtil
- GetSplitProjTransform() : Lighting::PSSMUtil
- GetStackSize() : Win360::Win360Thread, OSX::OSXThread, Win360::Win360Thread
- GetStageByName() : Graphics::GraphicsServer, InternalGraphics::InternalGraphicsServer
- GetStageName() : Graphics::View
- GetStartKeyIndex() : `CoreAnimation::AnimClip`
- GetStartTime() : `Animation::AnimJob`, `CoreAnimation::AnimClip`, `Animation::AnimJob`
- GetStateAsInputEvents() : `Base::GamePadBase`
- GetStateHandlerAt() : `App::GameApplication`
- GetStaticKey() : `CoreAnimation::AnimCurve`
- GetStatus() : `Http::HttpRequest`
- GetStatusCode() : `Http::HttpResponseReader`
- GetStream() : `IO::StreamWriter`, `Base::StreamTextureSaverBase`, `IO::StreamReader`, `IO::StreamWriter`, `IO::StreamReader`, `Base::StreamTextureSaverBase`, `IO::StreamReader`, `IO::StreamWriter`, `IO::StreamReader`, `IO::StreamWriter`, `IO::StreamReader`, `IO::StreamWriter`, `Net::DebugMessage`, `IO::StreamWriter`
- GetStreamByteOrder() : `IO::BinaryReader`, `IO::BinaryWriter`
- GetStreamClassByUriScheme() : `IO::SchemeRegistry`
- GetStreamVertexBuffer() : `Base::RenderDeviceBase`
- GetStreamVertexOffset() : `Base::RenderDeviceBase`
- GetStride() : `Jobs::TPJobCommand`
- GetString() : `Game::Entity`, `IO::XmlReader`, `Util::Variant`, `Script::ActionReader`, `BaseGameFeature::GlobalAttrsManager`, `BaseGameFeature::UserProfile`, `Util::CommandLineArgs`,
IO::BXmlReader
- GetStringArray() : Util::Variant
- GetStringLength() : Util::CommandLineArgs
- GetStringAttr() : Models::ModelNode
- GetSubstitution() : Script::SubstitutionManager
- GetSubSystemId() : CoreGraphics::AdapterInfo
- GetSubTasks() : Script::Task
- GetSubType() : IO::MediaType
- GetSurfaceInfo() : Resources::D3D9TextureStreamer
- GetSyncEvent() : Jobs::TPJobCommand
- GetSystemInfo() : Win32::SysFunc, OSX::SysFunc
- GetSystemTime() : Base::CalendarTimeBase, Win360::Win360CalendarTime
- GetTableIndex() : IO::ExcelXmlReader
- GetTableName() : IO::ExcelXmlReader
- GetTail() : IO::URI
- GetTakeById() : Script::Dialog
- GetTakeSoundID() : Script::Dialog, Script::DialogManager
- GetTakeSpeaker() : Script::Dialog, Script::DialogManager
- GetTargetEntityName() : Script::Task
- GetTargetResource() : Resources::LoadingResource
- GetTargetState() : FSM::Transition
- GetTaskViewed() : Script::Task
- GetTempDirectory() : Win360::Win360FSWrapper, OSX::OSXFSWrapper
- GetTemplateDataset() : BaseGameFeature::CategoryManager::Category
- GetTemplateTable() : BaseGameFeature::CategoryManager
- GetTemplateTableName() : BaseGameFeature::CategoryManager::Category
- GetText() : CoreGraphics::TextElement, Script::Task
- GetTextList() : Script::DialogTake
- GetTexture() : Resources::ManagedTexture
- GetTextureByIndex() : Frame::FrameShader
- GetTextureByName() : Frame::FrameShader
- GetTextureUnloaded() : Resources::ManagedTexture
- GetThenBlock() : Actions::IfThenElseAction
- GetThreadId() : CoreGraphics::TextElement, CoreGraphics::RenderShape
- `GetTicks()`: `BaseGameFeature::TimeSource`, `Win360::Win360Timer`, `BaseGameFeature::TimeSource`, `FrameSync::FrameSyncTimer`, `BaseGameFeature::TimeSource`
- `GetTime()`: `BaseGameFeature::TimeSource`, `FrameSync::FrameSyncTimer`, `BaseGameFeature::TimeSource`, `CoreAnimation::AnimEvent`, `BaseGameFeature::TimeSource`, `Animation::AnimSequencer`, `Win360::Win360Timer`, `App::RenderApplication`, `BaseGameFeature::TimeSource`, `App::RenderApplication`
- `GetTimeFactor()`: `Animation::AnimJob`, `FrameSync::FrameSyncTimer`, `BaseGameFeature::TimeManager`
- `GetTimeLeft()`: `Actions::Action`, `Actions::SequenceAction`
- `GetTimeOffset()`: `Animation::AnimJob`
- `GetTimeSourceByClassName()`: `BaseGameFeature::TimeManager`
- `GetTimeSourceByIndex()`: `BaseGameFeature::TimeManager`
- `GetTimeStamp()`: `Script::DialogTake`
- `GetTitle()`: `Script::Task`, `Base::GameContentServerBase`
- `GetTitleId()`: `Base::GameContentServerBase`
- `GetToByteOrder()`: `System::ByteOrder`
- `GetTopology()`: `CoreGraphics::RenderShape`
- `GetTrackIndex()`: `Animation::AnimJob`
- `GetTransitionAtIndex()`: `FSM::State`
- `GetTriggerPosition()`: `PhysicsFeature::TriggerProperty`
- `GetTriggerScale()`: `PhysicsFeature::TriggerProperty`
- **GetTriggerTransform**: `PhysicsFeature::TriggerProperty`
- **GetTypeRef**: `Visibility::ObserverContext`
- **GetUniformDesc**: `Base::JobBase`
- **GetUniqueId**: `Game::Entity`
- **GetUpVector**: `BaseGameFeature::EnvQueryManager`
- **GetURI**: `Resources::ResourceDictionary, IO::Stream, IO::ArchiveBase, IO::Stream, IO::ArchiveBase, Http::HttpRequest`
- **GetUsage**: `Base::ResourceBase, CoreGraphics::MemoryMeshLoader, Base::ResourceBase, Win360::D3D9StreamMeshLoader, Base::ResourceBase, Win360::D3D9StreamMeshLoader, Base::ResourceBase`
- **GetUseCount**: `Resources::Resource`
- **GetUsedMemory**: `Resources::PoolResourceMapper`
- **GetUserDirectory**: `OSX::OSXFSWrapper, Win360::Win360FSWrapper`
- **GetUserProfile**: `BaseGameFeature::LoaderServer`
- **GetVariableByIndex**: `Frame::FramePassBase, Frame::FrameBatch, Base::ShaderInstanceBase, Frame::FramePassBase`
- GetVariableByName() : Base::ShaderInstanceBase
- GetVariableBySemantic() : Base::ShaderInstanceBase
- GetVariationByFeatureMask() : Base::ShaderInstanceBase
- GetVariationByIndex() : Base::ShaderInstanceBase
- GetVariationCharJointComponents() : Characters::CharacterVariationSet
- GetVector() : Script::ActionReader
- GetVendorId() : CoreGraphics::AdapterInfo
- GetVersion() : Base::GameContentServerBase
- GetVertexBuffer() : Base::MeshBase
- GetVertexBufferByteSize() : Base::VertexBufferBase
- GetVertexBufferComponents() : Base::VertexBufferBase
- GetVertexBufferData() : CoreGraphics::RenderShape
- GetVertexLayout() : Direct3D9::D3D9ParticleRenderer, Base::VertexBufferBase, Direct3D9::D3D9ParticleRenderer, Base::VertexBufferBase, Base::RenderDeviceBase, Base::VertexBufferBase, Base::RenderDeviceBase
- GetVertexWidth() : CoreGraphics::RenderShape
- GetView() : Graphics::CameraEntity
- GetViewByName() : InternalGraphics::InternalGraphicsServer, Graphics::GraphicsServer
- GetViewClass() : Graphics::View
- GetViewFrustum() : InternalGraphics::InternalCameraEntity, Graphics::CameraEntity, Shared::CameraSettings
- GetViewProjTransform() : Shared::CameraSettings, Graphics::CameraEntity, Base::TransformDeviceBase, InternalGraphics::InternalCameraEntity, Base::TransformDeviceBase
- GetViewTransform() : Base::TransformDeviceBase, InternalGraphics::InternalCameraEntity, Graphics::CameraEntity, Base::TransformDeviceBase, Shared::CameraSettings
- GetVisibilityChecker() : InternalGraphics::InternalStage
- GetVisibilityContexts() : Visibility::VisibilityContainer
- GetVisibleEntities() : Visibility::VisibilityQuery
- GetVisibleFrameId() : Visibility::VisibilityContext
- GetVisibleModelNodeInstances() : Models::ModelNode,
Models::VisResolver, Models::ModelNode
- GetVisibleModelNodes(): Models::VisResolver
- GetVisibleModels(): Models::VisResolver
- GetVisualizeMipMaps(): Base::RenderDeviceBase
- GetWaitTimeout(): Messaging::BlockingHandlerThread
- GetWeekday(): Base::CalendarTimeBase
- GetWeight(): Animation::AnimEventInfo
- GetWidth(): Base::TextureBase, Base::RenderTargetBase, CoreGraphics::DisplayMode, Base::RenderTargetBase
- GetWindowTitle(): Base::DisplayDeviceBase, Graphics::DisplaySettings, Base::DisplayDeviceBase
-GetXPos(): CoreGraphics::DisplayMode
- GetYear(): Base::CalendarTimeBase
-GetYPos(): CoreGraphics::DisplayMode
- GetZFar(): Shared::CameraSettings
- GetZNear(): Shared::CameraSettings
- GlobalAttrsManager(): BaseGameFeature::GlobalAttrsManager
- GlobalLightEntity(): Graphics::GlobalLightEntity
- GlobalStringAtomTable(): Util::GlobalStringAtomTable
- Graphic: Script::InfoLog
- GraphicsEntity(): Graphics::GraphicsEntity
- GraphicsEntityShared(): Shared::GraphicsEntityShared
- GraphicsHandler(): Graphics::GraphicsHandler
- GraphicsInterface(): Graphics::GraphicsInterface
- GraphicsPageHandler(): Debug::GraphicsPageHandler
- GraphicsProperty(): GraphicsFeature::GraphicsProperty
- GraphicsServer(): Graphics::GraphicsServer
- greater3_all(): Math::float4
- greater3_any(): Math::float4
- greater4_all(): Math::float4
- greater4_any(): Math::float4
- greaterequal3_all(): Math::float4
- greaterequal3_any(): Math::float4
- greaterequal4_all(): Math::float4
- greaterequal4_any(): Math::float4
- gt(): Math::float2
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- h -

- Handle() : `Messaging::StaticMessageHandler`
- HandleAnimEvents() : `Animation::AnimEventServer`
- HandleCameraDistanceChange() : `GraphicsFeature::ChaseCameraProperty`
- HandleCameraOrbit() : `GraphicsFeature::ChaseCameraProperty`
- HandleCameraReset() : `GraphicsFeature::ChaseCameraProperty`
- HandleCharacterAnimDrivenMotion() : `InternalGraphics::InternalModelEntity`
- HandleCharacterAnimEvents() : `InternalGraphics::InternalModelEntity`
- HandleCommand() : `Commands::CreateEntityCommand`
- Handled() : `Messaging::Message`
- HandleDefaultEntity() : `Commands::CreateEntityCommand`
- HandleDeferredMessages() : `InternalGraphics::InternalGraphicsEntity`
- HandleDelayedJobs() : `BaseGameFeature::EntityManager`
- HandleEvent() : `Animation::AnimEventHandlerBase`, `Win32::Win32InputDisplayEventHandler`, `CoreGraphics::DisplayEventHandler`, `CoreGraphics::RenderEventHandler`, `CoreGraphics::ThreadSafeDisplayEventHandler`, `CoreGraphics::Win32InputDisplayEventHandler`
CoreGraphics::ThreadSafeRenderEventHandler
- HandleLeftMouseBtnDown() : PhysicsFeature::MouseGripperProperty
- HandleLeftMouseBtnUp() : PhysicsFeature::MouseGripperProperty
- HandleMoveDirection() : PhysicsFeature::ActorPhysicsProperty
- HandleMoveFollow() : PhysicsFeature::ActorPhysicsProperty
- HandleMoveGoto() : PhysicsFeature::ActorPhysicsProperty
- HandleMoveRotate() : PhysicsFeature::ActorPhysicsProperty
- HandleMoveTurn() : PhysicsFeature::ActorPhysicsProperty
- HandlePendingEvents() :
  CoreGraphics::ThreadSafeDisplayEventHandler ,
  CoreGraphics::ThreadSafeRenderEventHandler ,
  CoreGraphics::ThreadSafeDisplayEventHandler
- HandlePendingRequests() : Http::HttpRequestHandler ,
  Http::HttpServerProxy , Http::HttpRequestHandler
- Handler() : Messaging::Handler
- HandleRequest() : Debug::StreamingTexturePageHandler ,
  Debug::ObjectInspectorHandler ,
  Debug::HelloWorldRequestHandler ,
  Debug::MemoryPageHandler , Debug::IoPageHandler ,
  Debug::ThreadPageHandler , Debug::DisplayPageHandler ,
  Debug::MeshPageHandler , Debug::ShaderPageHandler ,
  Debug::TexturePageHandler , Debug::DebugPageHandler ,
  Debug::SvgTestPageHandler , Debug::CorePageHandler ,
  Debug::GraphicsPageHandler , Debug::ConsolePageHandler ,
  Http::DefaultHttpRequestHandler ,
  Http::HttpRequestHandler
- HandlerThreadBase() : Messaging::HandlerThreadBase
- HandleSetTransform() :
  PhysicsFeature::ActorPhysicsProperty
- HandleTrackedJoints() : InternalGraphics::InternalModelEntity
- HasActionScript() : Script::ScriptManager
- HasActiveEntities() : BaseGameFeature::EntityManager
- HasArg() : Util::CommandLineArgs
- HasAssign() : IO::AssignRegistry
- HasAttr() : BaseGameFeature::UserProfile ,
  BaseGameFeature::GlobalAttrsManager , IO::XmlReader ,
  IO::BXmlReader , Game::Entity
- HasCategory() : BaseGameFeature::CategoryManager ,
  CoreAnimation::AnimEvent
- HasCharacter() : InternalGraphics::InternalModelEntity
- HasChild() : Models::ModelNode ,
  Models::ModelNodeInstance , Models::ModelNode ,
  Models::ModelNodeInstance , Models::ModelNode ,
  Models::ModelNodeInstance , Models::ModelNode ,
  Models::ModelNodeInstance , Models::ModelNode ,
  Models::ModelNodeInstance , Models::ModelNode
- HasClip() : CoreAnimation::AnimResource
- HasColumn() : `Util::SparseTable< TYPE >`, `IO::ExcelXmlReader`
- HasComponent() : `Base::VertexLayoutBase`
- HasCondition() : `Script::DialogTake`
- HasContent() : `IO::XmlReader`
- HasCPUResolveTexture() : `Base::RenderTargetBase`
- HasData() : `Net::DebugPacket`
- HasDepthResolveTexture() : `Base::RenderTargetBase`
- HasDepthStencilBuffer() : `Base::RenderTargetBase`
- HasDialogs() : `Script::DialogManager`
- HasElseBlock() : `Actions::IfThenElseAction`
- HasEntity() : `Game::Property`, `Conditions::Condition`, `InternalGraphics::InternalGraphicsServer`, `Game::Property`, `Conditions::Condition`, `Game::Property`, `Conditions::Condition`, `Game::Property`, `Actions::Action`, `Game::Property`, `Actions::Action`, `Game::Property`, `Actions::Action`, `Game::Property`
- HasEntry() : `Resources::ResourceDictionary`
- HasEvent() : `CoreAnimation::AnimClip`
- HasFocus() : `GraphicsFeature::CameraProperty`
- HasFramePassBase() : `Frame::FrameShader`
- HashCode() : `Util::String`, `Win32::Win32Guid`, `Util::Blob`, `OSX::OSXGuid`
- HashTable() : `Util::HashTable< KEYTYPE, VALUETYPE >`
- HasIndexBuffer() : `Base::MeshBase`
- HasInfo() : `Script::InfoLog`
- HasInfoInCurrentSection() : `Script::InfoLog`
- HasInput() : `OSX::OSXConsoleHandler`, `Win32::Win32ConsoleHandler`, `IO::ConsoleHandler`, `IO::Console`, `IO::ConsoleHandler`
- HasInstanceDataset() : `BaseGameFeature::CategoryManager::Category`
- HasInstanceTable() : `BaseGameFeature::CategoryManager`
- HasLightEntity() : `GraphicsFeature::LightFlickerUtil`
- HasManagedModel() : `Models::ModelServer`
- HasManagedResource() : `Resources::ResourceManager`
- HasMapper() : `Resources::ResourceManager`
- HasMessages() : `Net::TcpMessageCodec`
- HasMouseIntersection() :
- `BaseGameFeature::EnvQueryManager`
  - `HasMultipleRenderTarget()`: `Frame::FrameShader`
  - `HasNextTakeLayer()`: `Script::Dialog`
  - `HasNode()`: `IO::XmlReader`, `IO::BXmlReader`
  - `HasOpenConditions()`: `Script::Task`
  - `HasParent()`: `Models::ModelNode`, `Util::SimpleTree<VALUETYPE>::Node`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`
  - `HasParentJoint()`: `Characters::CharacterJoint`
  - `HasPassRenderTarget()`: `Base::RenderDeviceBase`
  - `HasPort()`: `Messaging::Dispatcher`
  - `HasRecvData()`: `Win360::Win360Socket`
  - `HasRenderDebug()`: `Game::FeatureUnit`
  - `HasRenderTarget()`: `InternalGraphics::InternalView`, `Frame::FrameShader`
  - `HasResolveTexture()`: `Base::RenderTargetBase`
  - `HasResource()`: `Resources::ResourceManager`
  - `HasRow()`: `Util::SparseTable<TYPE>`
  - `HasShader()`: `Base::ShaderServerBase`
  - `HasShaderVariableInstance()`: `Models::StateNodeInstance`
  - `HasShapes()`: `PhysicsFeature::EnvironmentCollideProperty`
  - `HasSharedVariableBySemantic()`: `Base::ShaderServerBase`
  - `HasSkin()`: `Characters::CharacterSkinLibrary`, `Characters::CharacterSkinSet`
  - `HasSkinList()`: `Characters::CharacterSkinLibrary`
  - `HasStarted()`: `Game::GameServer`
  - `HasStream()`: `IO::StreamWriter`, `IO::StreamReader`, `IO::StreamWriter`, `IO::StreamReader`, `IO::StreamWriter`, `IO::StreamReader`, `IO::StreamWriter`, `IO::StreamReader`, `IO::StreamWriter`, `IO::StreamReader`, `IO::StreamWriter`
  - `HasStringAttr()`: `Models::ModelNode`
  - `HasSubTasks()`: `Script::Task`
  - `HasTaskStatusChanged()`: `Script::Task`
- HasTemplateDataset(): BaseGameFeature::CategoryManager::Category
- HasTemplateTable(): BaseGameFeature::CategoryManager
- HasTexture(): Frame::FrameShader
- HasVariableByName(): Base::ShaderInstanceBase
- HasVariableBySemantic(): Base::ShaderInstanceBase
- HasVariation(): Base::ShaderInstanceBase
- HasVertexBuffer(): Base::MeshBase
- HeaderSize: Net::DebugPacket
- Height(): Util::FixedTable< TYPE >
- height(): Math::rectangle< TYPE >
- HelloWorldRequestHandler(): Debug::HelloWorldRequestHandler
- hermite(): Math::float4
- HistoryConsoleHandler(): IO::HistoryConsoleHandler
- HoldResources(): Resources::ResourceManager
- HorizontalRule(): Http::HtmlPageWriter
- Host(): IO::URI
- HtmlPageWriter(): Http::HtmlPageWriter
- HttpInterface(): Http::HttpInterface
- HttpMessageHandler(): Http::HttpMessageHandler
- HttpRequest(): Http::HttpRequest
- HttpRequestHandler(): Http::HttpRequestHandler
- HttpRequestReader(): Http::HttpRequestReader
- HttpRequestWriter(): Http::HttpRequestWriter
- HttpResponseReader(): Http::HttpResponseReader
- HttpServer(): Http::HttpServer
- HttpServerProxy(): Http::HttpServerProxy
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- i -

- Id : `InternalGraphics::InternalGraphicsEntity`, `Messaging::Id`, `InternalGraphics::InternalGraphicsEntity`
- identity() : `Math::matrix44`, `Math::quaternion`
- IfThenElseAction() : `Actions::IfThenElseAction`
- InbetweenTicks() : `CoreAnimation::AnimUtil`
- Incr() : `Debug::DebugCounter`
- IncrClientCount() : `Resources::ManagedResource`
- Increment() : `OSX::OSXInterlocked`, `Win360::Win360Interlocked`
- IncrUseCount() : `Resources::Resource`
- IndexBufferBase() : `Base::IndexBufferBase`
- IndexTypeAsD3D9Format() : `Win360::D3D9Types`
- InfoLog() : `Script::InfoLog`
- Init() : `Actions::Action`, `Actions::ActionList`, `Actions::Action`
- InitNetwork() : `Win360::Win360Socket`
- InputEvent() : `Input::InputEvent`
- InputHandler() : `Input::InputHandler`
- InputServer() : `Input::InputServer`
- InputServerBase() : `Base::InputServerBase`
- InputTimeSource() : `BaseGameFeature::InputTimeSource`
- Insert() : `Util::SimpleTree< VALUETYPE >::Node`, `Util::Array< TYPE >`, `Threading::SafePriorityQueue< PRITYPE, TYPE >`, `Util::Array< TYPE >`
- InsertAtEndOfIdenticalRange() : `Util::Array< TYPE >`
- InsertSorted() : `Util::Array< TYPE >`
- InsertVisibilityContainer() : `Visibility::VisibilityBoxSystem`, `Visibility::VisibilityClusterSystem`, `Visibility::VisibilitySystemBase`
- InsertVisibilityContext() : `Visibility::VisibilityBoxSystem`, `Visibility::VisibilityClusterSystem`, `Visibility::VisibilitySystemBase`, `Visibility::VisibilityQuadtree`
- inside() : `Math::rectangle< TYPE >`, `Math::sphere`, `Math::frustum`
- Instance() : `Core::Factory`
- InterfaceBase() : `Interface::InterfaceBase`
- InterfaceHandlerBase() : `Interface::InterfaceHandlerBase`
- InternalAbstractLightEntity() : `Lighting::InternalAbstractLightEntity`
- InternalCameraEntity() : `InternalGraphics::InternalCameraEntity`
- InternalGlobalLightEntity() : `Lighting::InternalGlobalLightEntity`
- InternalModelEntity() : `InternalGraphics::InternalModelEntity`
- InternalPointLightEntity() : `Lighting::InternalPointLightEntity`
- InternalSpotLightEntity() : `Lighting::InternalSpotLightEntity`
- InternalStage() : `InternalGraphics::InternalStage`
- InternalView() : `InternalGraphics::InternalView`
- intersect() : `Math::line`
- intersect_sweep() : `Math::sphere`
- intersectline() : `Math::plane`
- intersects() : `Math::bbox`, `Math::sphere`
- Invalidate() : `Threading::ObjectRef`
- InvalidCoreId : `OSX::OSXCpu`, `Win32::Win32Cpu`
- InvalidModelNodeType : `Models::ModelNodeType`
- inverse() : `Math::matrix44`, `Math::quaternion`
- Invoke() : `Messaging::DelegateTable`
- IoInterfaceHandler() : `IO::IoInterfaceHandler`
- IoPageHandler() : `Debug::IoPageHandler`
- IoServer() : `IO::IoServer`
- `IsA() : Core::RefCounted`
- `IsActive() : Game::Manager, Game::Property, Game::Manager, Game::Property, Game::Manager, Game::Property, Game::FeatureUnit, Game::Property, Game::Entity, Game::FeatureUnit, Game::Property, Animation::AnimJob, Game::Manager, Game::Property, Game::Manager, Game::Property, Game::Manager, Game::Property, Script::Dialog, InternalGraphics::InternalGraphicsEntity, Game::Property, InternalGraphics::InternalGraphicsEntity, CoreAnimation::AnimCurve, InternalGraphics::InternalGraphicsEntity, Animation::AnimJob, Game::Property, Game::FeatureUnit, Game::Manager, Game::Property, Game::Manager, Game::Property, Game::Manager, Game::FeatureUnit, Game::Manager, InternalGraphics::InternalGraphicsEntity, Game::Property, Game::Manager`
- `IsAlNum() : Util::String`
- `IsAlpha() : Util::String`
- `IsAlwaysOnTop() : Base::DisplayDeviceBase, Graphics::DisplaySettings, Base::DisplayDeviceBase`
- `IsAnimDrivenMotionEnabled() : Characters::CharacterAnimationController`
- `IsAnimDrivenMotionTrackingEnabled() : Graphics::ModelEntity, InternalGraphics::InternalModelEntity`
- `IsAnimEventTrackingEnabled() : Graphics::ModelEntity, InternalGraphics::InternalModelEntity`
- `IsArchiveFileSystemEnabled() : IO::IoServer`
- `IsArchiveMounted() : IO::IoServer`
- `IsAttached() : Input::InputHandler`
- `IsAttachedToModel() : Models::ModelNode`
- `IsAttachedToResource() : Resources::ResourceLoader, Resources::ResourceSaver, Resources::ResourceLoader, Resources::ResourceSaver, Resources::ResourceLoader`
- `IsAttachedToResourceManager() : Resources::ResourceMapper`
- `IsAttachedToSequencer() : Animation::AnimJob`
- `IsAttachedToServer() : InternalGraphics::InternalStage`,

**InternalGraphics::InternalView**
- `IsAttachedToStage()`: `InternalGraphics::InternalGraphicsEntity`
- `IsAttachedToView()`: `InternalGraphics::InternalCameraEntity, Graphics::CameraEntity`
- `IsAutoManaged()`: `Resources::ManagedResource`
- `IsBlocking()`: `Net::StdTcpClient`
- `IsBound()`: `Win360::Win360Socket`
- `IsCapturing()`: `Input::InputHandler`
- `IsCharJointDataValid()`: `Graphics::ModelEntity, InternalGraphics::InternalModelEntity`
- `IsCharJointTrackingEnabled()`: `Graphics::ModelEntity, InternalGraphics::InternalModelEntity`
- `IsConnected()`: `Base::GamePadBase, Net::StdTcpClientConnection, Net::StdTcpClient, Net::StdTcpClientConnection, Win360::Win360Socket, Base::GamePadBase`
- `IsConversation()`: `Script::DialogDesc`
- `IsDebugHudEnabled()`: `Animation::AnimSequencer`
- `IsDefaultRenderTarget()`: `Base::RenderTargetBase`
- `IsDefaultView()`: `Graphics::View`
- `IsDeferred()`: `Messaging::Message`
- `IsDerivedFrom()`: `Core::Rtti`
- `IsDeviceName()`: `OSX::OSXFSWrapper, Win360::Win360FSWrapper`
- `IsDialogActive()`: `Script::DialogManager`
- `IsDialogLoaded()`: `Script::DialogManager`
- `IsDigit()`: `Util::String`
- `isdirty()`: `Math::transform44`
- `IsDisplayModeSwitchEnabled()`: `Graphics::DisplaySettings, Base::DisplayDeviceBase`
- `IsEmpty()`: `Util::Queue<TYPE>, Util::RingBuffer<TYPE>, Util::Array<TYPE>, Util::SimpleTree<VALUETYPE>::Node, Util::Stack<TYPE>, Util::String, Threading::SafeQueue<TYPE>, Util::List<TYPE>, Util::Array<TYPE>, Threading::SafePriorityQueue<PRITYPE, TYPE>, IO::URI, Util::Dictionary<KEYTYPE, VALUETYPE>, Util::FixedArray<TYPE>, Util::HashTable<KEYTYPE, VALUETYPE>, Util::List<TYPE>`
- `IsEnabled()`: `PhysicsFeature::PhysicsProperty`
GraphicsFeature::LightFlickerUtil
- IsEntityInDelayedJobs() : BaseGameFeature::EntityManager
- IsExpired() : Animation::AnimJob
- IsFeatureAttached() : Game::GameServer
- IsFinalPacket() : Net::DebugPacket
- IsFinished() : Visibility::VisibilityQuery
- IsFixedFrameTime() : FrameSync::FrameSyncHandlerThread
- IsFullscreen() : Base::DisplayDeviceBase, Graphics::DisplaySettings, Base::DisplayDeviceBase
- IsGotoActive() : PhysicsFeature::ActorPhysicsProperty
- isidentity() : Math::matrix44, Math::quaternion
- IsInAttach() : Base::ParticleRendererBase
- IsInBeginFrame() : Base::RenderDeviceBase
- IsInfinite() : Animation::AnimJob
- IsInFocus() : BaseGameFeature::EntityManager
- InstanceOf() : Core::RefCounted
- IsInViewSpace() : Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance
- IsLoaded() : BaseGameFeature::UserProfile, Resources::Resource, Script::Dialog, Script::Task, Resources::Resource
- IsLoading() : BaseGameFeature::EntityLoaderBase
- IsLocked() : Script::DialogDesc, Resources::Resource
- IsLower() : Util::String
- IsMapped() : IO::Stream, CoreAnimation::AnimKeyBuffer, IO::Stream
- IsMarkedForRemove() : InternalGraphics::InternalGraphicsEntity
- IsMemoryMappingEnabled() : IO::BinaryReader, IO::BinaryWriter
- IsNetworkInitialized() : Win360::Win360Socket
- IsNull() : Util::BitField< NUMBITS >
- IsObjectRefValid() : Graphics::GraphicsEntity
- IsOpen() : App::Application, IO::StreamWriter, Models::ModelServer, IO::StreamWriter, App::Application, IO::StreamWriter, Base::InputServerBase,
Base::ShapeRendererBase, Frame::FrameServer,
IO::ConsoleHandler, Resources::ResourceManager,
IO::StreamReader, IO::Stream, Base::TextRendererBase,
Messaging::AsyncPort, Base::ShaderServerBase,
Messaging::Handler, Core::CoreServer,
Base::DisplayDeviceBase, IO::Stream, Messaging::Handler,
IO::StreamReader, Base::TransformDeviceBase,
Messaging::Handler, Lighting::LightServerBase,
IO::StreamReader, Lighting::LightServerBase,
Base::InputServerBase, Visibility::VisibilitySystemBase,
Base::DisplayDeviceBase, App::Application,
Messaging::Handler, Animation::AnimEventServer,
Lighting::LightServerBase, IO::StreamWriter,
IO::ConsoleHandler, Visibility::VisibilitySystemBase,
Base::InputServerBase, IO::StreamWriter, IO::Stream,
Messaging::Handler, Graphics::GraphicsServer,
App::Application, Graphics::Display, IO::ConsoleHandler,
Base::ShaderServerBase, http::HttpServer,
IO::StreamWriter, InternalGraphics::InternalGraphicsServer,
Base::DisplayDeviceBase, Net::StdTcpServer,
Base::VertexLayoutServerBase, Lighting::LightServerBase,
Base::TransformDeviceBase, Messaging::Handler,
Base::RenderDeviceBase, Base::VertexLayoutServerBase,
IO::StreamReader, Messaging::AsyncPort,
IO::StreamReader, Http::HttpServerProxy,
Visibility::VisibilitySystemBase, Base::TextRendererBase,
Models::VisResolver, Debug::DebugServer,
Base::DisplayDeviceBase,
InternalGraphics::AttachmentServer, Messaging::AsyncPort,
Base::RenderDeviceBase, Messaging::Handler,
Messaging::AsyncPort, IO::StreamReader, IO::Console,
Base::RenderDeviceBase, IO::ConsoleHandler,
Base::TextRendererBase, Net::StdTcpServer,
Win360::Win360Socket, Messaging::AsyncPort,
Win360::Win360Socket, Messaging::Handler,
FSM::StateMachine, Base::ShapeRendererBase, IO::Stream,
IO::StreamWriter, Particles::ParticleServer,
BaseGameFeature::LoaderServer, App::Application,
Messaging::AsyncPort, Visibility::VisibilitySystemBase,
Base::ShapeRendererBase, Base::TransformDeviceBase, IO::ConsoleHandler
- IsOrthogonal() : Shared::CameraSettings
- IsPaused() : BaseGameFeature::TimeSource
- IsPending() : Resources::Resource, Animation::AnimJob, Resources::Resource, Animation::AnimJob, Resources::Resource
- IsPerfHUDEnabled() : Debug::DebugGraphicsHandler
- IsPerspective() : Shared::CameraSettings
- IsPlaceholder() : Resources::ManagedResource
- ispointinside() : Math::matrix44
- IsPoolBlock() : Win360::Win360MemoryPool, OSX::OSXMemoryPool
- IsQuitRequested() : Base::InputServerBase, App::RenderApplication, Base::InputServerBase, Game::GameServer, Base::InputServerBase, App::RenderApplication
- IsReadOnly() : IO::IoServer, Win360::Win360FSWrapper, OSX::OSXFSWrapper
- IsResolved() : Models::VisResolveContainer< TYPE >
- IsRunning() : Win360::Win360Thread, OSX::OSXThread, Win360::Win360Thread
- IsSignalOnEnqueueEnabled() : Threading::SafeQueue< TYPE >
- IsSingleThreadMode() : Http::HttpServer
- IsSorted() : Util::Array< TYPE >
- IsSpeaker() : Script::Dialog
- IsSpecial() : BaseGameFeature::CategoryManager::Category
- IsStatic() : CoreAnimation::AnimCurve
- IsStoppingOrExpired() : Animation::AnimJob
- IsTimeEffectActive() : BaseGameFeature::TimeManager
- IsTimeRunning() : FrameSync::FrameSyncTimer
- IsTriggerActive() : PhysicsFeature::TriggerProperty
- IsTripleBufferingEnabled() : Base::DisplayDeviceBase, Graphics::DisplaySettings, Base::DisplayDeviceBase
- IsUpper() : Util::String
- IsUriSchemeRegistered() : IO::SchemeRegistry
- IsValid() : Characters::CharacterSkinSet, Base::ShaderInstanceBase, OSX::OSXGuid, Base::JobBase, InternalGraphics::InternalGraphicsEntity,
isValid() : Ptr< TYPE >
IsValid() : Particles::ParticleSystem, Base::JobBase, CoreAnimation::AnimSampleBuffer, IO::SchemeRegistry, Base::GameContentServerBase, Models::ModelNodeInstance, Base::ParticleRendererBase, Particles::EmitterMesh, Base::ShaderInstanceBase, InternalGraphics::InternalGraphicsEntity, Base::JobPortBase, Util::String, Graphics::GraphicsEntity, Base::SkinnedMeshRendererBase
isValid() : WeakPtr< TYPE >
IsValid() : Animation::AnimSequencer, Util::Blob, Characters::CharacterServer, Win32::Win32Guid, InternalGraphics::InternalGraphicsEntity, Models::ModelNodeInstance, Base::JobBase, Models::ModelNodeInstance, Base::JobPortBase, Characters::Character, Util::StringBuffer, BaseGameFeature::CategoryManager::Entry, Net::DebugMessage, Base::ParticleRendererBase, Characters::CharacterInstance, Debug::DebugTimer, Base::MouseRenderDeviceBase, Graphics::GraphicsEntity, IO::ArchiveBase, RenderModules::RTPluginRegistry, Base::GameContentServerBase, Characters::CharacterVariationSet, Particles::EnvelopeSampleBuffer, CoreAnimation::AnimKeyBuffer, IO::ArchiveBase,
Win360::Win360ThreadBarrier, Base::RenderTargetBase, Base::SkinnedMeshRendererBase, InternalGraphics::InternalGraphicsEntity, Graphics::View, IO::MediaType, Base::JobBase, FrameSync::FrameSyncTimer, CoreGraphics::RenderShape, IO::ArchiveBase, Resources::ResourceDictionary, Base::JobPortBase, Graphics::GraphicsEntity, Base::ShaderInstanceBase, Characters::CharacterSkeleton, Base::ParticleRendererBase, RenderModules::RenderModule

isValid() : OSX::OSXThreadLocalPtr<TYPE>
IsValid() : Util::StringAtom, RenderUtil::DrawFullScreenQuad, Base::RenderTargetBase, Base::VertexLayoutBase
IsValidBool() : Util::String
IsValidFloat() : Util::String
IsValidFloat2() : Util::String
IsValidFloat4() : Util::String
IsValidForRendering() : Characters::CharacterInstance
IsValidHttpRequest() : Http::HttpRequestReader
IsValidHttpResponse() : Http::HttpResponseReader
IsValidInt() : Util::String
IsValidMatrix44() : Util::String
IsVerticalSyncEnabled() : Graphics::DisplaySettings, Base::DisplayDeviceBase
IsVirtual() : BaseGameFeature::CategoryManager::Category
InternalGraphics::InternalGraphicsEntity

- Iterator: Util::Array< TYPE >, Util::List< TYPE >::Iterator, Util::List< TYPE >::Iterator
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_ 
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- j -

- JobBase() : Base::JobBase
- JobDataDesc() : Jobs::JobDataDesc
- JobPortBase() : Base::JobPortBase
- JobSystem() : Jobs::JobSystem
- JobUniformDesc() : Jobs::JobUniformDesc
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **k** -

- Key() : `Util::KeyValuePair< KEYTYPE, VALUETYPE >`
- KeyAtIndex() : `Util::Dictionary< KEYTYPE, VALUETYPE >`
- KeyboardBase() : `Base::KeyboardBase`
- KeyCodesByGroup() : `Input::Key`
- KeyDown() : `Base::KeyboardBase`
- KeyPressed() : `Base::KeyboardBase`
- KeysAs() : `Util::Dictionary< KEYTYPE, VALUETYPE >`
- KeysAsArray() : `Util::Dictionary< KEYTYPE, VALUETYPE >`
- KeyUp() : `Base::KeyboardBase`
- KeyValuePair() : `Util::KeyValuePair< KEYTYPE, VALUETYPE >`
- KeyValuePairAtIndex() : `Util::Dictionary< KEYTYPE, VALUETYPE >`
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Enumerator
- Related Functions

- _
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- l -
  - le() : Math::float2
  - Leave() : OSX::OSXCriticalSection, Win360::Win360CriticalSection
  - LeaveLockStepMode() :
    FrameSync::FrameSyncHandlerThread, Graphics::GraphicsInterface
  - length() : Math::float2, Math::quaternion, Math::line, Math::float4
  - Length() : Util::String
  - lengthsq() : Math::float2, Math::line, Math::quaternion, Math::float4
  - lerp() : Math::float4
  - less3_all() : Math::float4
  - less3_any() : Math::float4
  - less4_all() : Math::float4
  - less4_any() : Math::float4
  - lessequal3_all() : Math::float4
  - lessequal3_any() : Math::float4
  - lessequal4_all() : Math::float4
  - lessequal4_any() : Math::float4
  - Level() : Util::QuadTree< TYPE >::Node
  - LevelExists() : BaseGameFeature::CategoryManager
  - LightFlickerUtil() : GraphicsFeature::LightFlickerUtil
- LightPrePassServer() : Lighting::LightPrePassServer
- LightServer() : Lighting::LightServer
- LightServerBase() : Lighting::LightServerBase
- Line() : Http::SvgPageWriter
- line() : Math::line
- Line() : Http::SvgPageWriter
- LineBreak() : Http::HtmlPageWriter
- LinkType : InternalGraphics::InternalGraphicsEntity
- List() : Util::List< TYPE >
- ListDirectories() : IO::IoServer, IO::ZipArchive, OSX::OSXFSWrapper, IO::ZipArchive, IO::ArchiveBase, Win360::Win360FSWrapper
- Listen() : Win360::Win360Socket
- ListFiles() : IO::ZipArchive, OSX::OSXFSWrapper, IO::ArchiveBase, Win360::Win360FSWrapper, IO::IoServer, IO::ZipArchive
- ln() : Math::quaternion
- load() : Math::float4
- Load() : Resources::Resource
- load() : Math::matrix44
- Load() : Resources::Resource, Resources::ResourceDictionary, Resources::Resource
- load() : Math::quaternion
- Load() : Resources::Resource, BaseGameFeature::EntityLoader, Resources::Resource, BaseGameFeature::EntityLoaderBase, Resources::Resource, BaseGameFeature::EnvironmentLoader, BaseGameFeature::LevelLoader, Resources::Resource, BaseGameFeature::UserProfile
- load() : Math::float4
- Load() : Resources::Resource, Script::Task
- load_float3() : Math::float4
- load_ubyte4n_signed() : Math::float4
- LoadActionFromScript() : Script::ScriptManager
- LoadActionScripts() : Script::ScriptManager
- LoadAssociatedDialogs() : Script::DialogManager
- LoadAttributes() : BaseGameFeature::GlobalAttrsManager
- LoadCanScript() : Script::ScriptManager
- LoadCloseActions() : Script::Task
- LoadCloseConditions() : Script::Task
- LoadCondition() : Script::ScriptManager
- LoadConditionScripts() : Script::ScriptManager
- LoadDialogContent() : Script::Dialog
- LoadDialogs() : Script::DialogManager
- LoadDialogTakeStates() : Script::DialogManager
- LoadEntities() : BaseGameFeature::LoaderServer
- LoaderServer() : BaseGameFeature::LoaderServer
- LoadFailActions() : Script::Task
- LoadFailConditions() : Script::Task
- LoadFailed() : Resources::Resource
- LoadFrameShader() : Frame::FrameShaderLoader
- LoadingResource() : Resources::LoadingResource
- LoadInstances() : BaseGameFeature::CategoryManager
- LoadLevel() : BaseGameFeature::LoaderServer
- LoadManagedModel() : Models::ModelServer
- LoadOnScript() : Script::ScriptManager
- LoadOpenActions() : Script::Task
- LoadOpenConditions() : Script::Task
- LoadResources() : Models::ModelNode, Models::Model, Models::ModelNode, Models::ShapeNode, Particles::ParticleSystemNode, Models::StateNode, Characters::CharacterNode
- LoadState() : Script::ScriptManager
- LoadStateMachine() : Script::ScriptManager
- LoadStateMachines() : Script::ScriptManager
- LoadStatement() : Script::ScriptManager
- LoadStates() : Script::ScriptManager
- LoadSubTasks() : Script::Task
- LoadTableContent() : Script::ScriptManager
- LoadTransition() : Script::ScriptManager
- LoadTransitions() : Script::ScriptManager
- loadu() : Math::float4, Math::matrix44, Math::quaternion, Math::float4
- LocalPath() : IO::URI
- LocalStringAtomTable() : Util::LocalStringAtomTable
- LocalTimeToFileTime() : Base::CalendarTimeBase, Win360::Win360CalendarTime
- Lock() : Resources::Resource
- LockStepModeActive()
  FrameSync::FrameSyncHandlerThread
- LockSurfaces() : Resources::D3D9TextureStreamer
- LodDistancesUsed() : Models::TransformNode
- LogFileConsoleHandler() : IO::LogFileConsoleHandler
- LogLevelToString() : Script::InfoLog
- lookatlh() : Math::matrix44
- lookatrh() : Math::matrix44
- LookupAnimatorNodeInstance() : RenderUtil::NodeLookupUtil
- LookupChild() : Models::ModelNode,
  Models::ModelNodeInstance, Models::ModelNode,
  Models::ModelNodeInstance, Models::ModelNode,
  Models::ModelNodeInstance, Models::ModelNode
- LookupElements() : Util::SparseTable< TYPE >
- LookupFrameShader() : Frame::FrameServer
- LookupManagedModel() : Models::ModelServer
- LookupManagedResource() : Resources::ResourceManager
- LookupNode() : Models::Model
- LookupNodeInstance() : Models::ModelInstance
- LookupPath() : Models::ModelNodeInstance
- LookupProjectDirectory() : App::RenderApplication
- LookupResource() : Resources::ResourceManager
- LookupSamples() : Particles::EnvelopeSampleBuffer
- LookupStateNodeInstance() : RenderUtil::NodeLookupUtil
- lt() : Math::float2
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **m** -

- managedResource : Resources::LoadingResource
- ManagedResource() : Resources::ManagedResource
- managedResource : Resources::LoadingResource
- Manager() : Game::Manager
- Map() : IO::MemoryStream, IO::FileStream, IO::Stream, Win360::D3D9IndexBuffer, CoreAnimation::AnimKeyBuffer, Base::TextureBase, IO::ZipFileStream, Base::IndexBufferBase, Direct3D9::D3D9Texture, Base::TextureBase, Win360::D3D9VertexBuffer, Base::TextureBase, Base::VertexBufferBase, Direct3D9::D3D9Texture, Win360::D3D9IndexBuffer, Win360::D3D9VertexBuffer
- MapCubeFace() : Base::TextureBase, Direct3D9::D3D9Texture, Base::TextureBase, Direct3D9::D3D9Texture
- MapInfo() : Base::TextureBase::MapInfo
- MapSampleCounts() : CoreAnimation::AnimSampleBuffer
- MapSamples() : CoreAnimation::AnimSampleBuffer
- MarkRemove() : InternalGraphics::InternalGraphicsEntity
- Mask : CoreGraphics::ShaderFeature
- MatchPattern() : Util::String
- matrix44() : Math::matrix44
- maximize() : Math::float2, Math::float4
- MaxNumModelNodeType: Models::ModelNodeType
- MaxNumRenderTargets: Base::MultipleRenderTargetBase
- MaxNumVertexStreams: Base::RenderDeviceBase
- MaxPayloadSize: Net::DebugPacket
- MaxSliceSize: Base::JobBase
- MayaCameraProperty(): GraphicsFeature::MayaCameraProperty
- MayaCameraUtil(): RenderUtil::MayaCameraUtil
- MediaType(): IO::MediaType
- MemoryIndexBufferLoaderBase(): Base::MemoryIndexBufferLoaderBase
- MemoryMeshLoader(): CoreGraphics::MemoryMeshLoader
- MemoryPageHandler(): Debug::MemoryPageHandler
- MemoryStream(): IO::MemoryStream
- MemoryVertexBufferLoaderBase(): Base::MemoryVertexBufferLoaderBase
- MeshBase(): Base::MeshBase
- MeshPageHandler(): Debug::MeshPageHandler
- Message(): Messaging::Message
- MessageBox(): OSX::SysFunc, Win32::SysFunc
- MessageClient(): Net::MessageClient
- MessageReader(): Messaging::MessageReader
- MessageWriter(): Messaging::MessageWriter
- minimize(): Math::float4, Math::float2
- Model(): Models::Model
- ModelEntity(): Graphics::ModelEntity
- ModelEntityShared(): Shared::ModelEntityShared
- ModelInstance(): Models::ModelInstance
- ModelNode(): Models::ModelNode
- ModelNodeInstance(): Models::ModelNodeInstance
- ModelServer(): Models::ModelServer
- ModulateStepTime(): Particles::ParticleSystem
- Month: Base::CalendarTimeBase
- MonthToString(): Base::CalendarTimeBase
- MountArchive(): IO::IoServer
- MountStandardArchives(): IO::IoServer
- MouseBase(): Base::MouseBase
- MouseGripperProperty(): PhysicsFeature::MouseGripperProperty
- MousePointer() : `CoreGraphics::MousePointer`
- MouseRenderDevice() : `CoreGraphics::MouseRenderDevice`
- MouseRenderDeviceBase() : `Base::MouseRenderDeviceBase`
- MouseRenderer() : `Graphics::MouseRenderer`
- MultipleRenderTargetBase() : `Base::MultipleRenderTargetBase`
- multiply() : `Math::matrix44`, `Math::float4`, `Math::quaternion`, `Math::float4`
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Enumerator
- Related Functions

- _
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- n -

- **Name**: Base::ShaderVariableBase, Models::ModelNodeType, Base::ShaderVariableBase, CoreGraphics::ShaderFeature, Base::ShaderVariableBase
- **NavigateToNextLayer()**: Script::Dialog
- **nearequal3()**: Math::float4
- **nearequal4()**: Math::float4
- **NeedsLightModelLinking()**: Lighting::LightServerBase, Lighting::LightPrePassServer, Lighting::LightServerBase, Lighting::LightPrePassServer
- **newSlot**: Resources::PoolLoadingResource
- **Next()**: Core::ExitHandler
- **NextString()**: Util::StringBuffer
- **NextToken()**: Script::ActionReader
- **Node()**: Util::QuadTree<TYPE>::Node, Util::SimpleTree<VALUETYPE>::Node
- **Non**: Script::InfoLog
- **normalize()**: Math::float2, Math::float4, Math::plane, Math::float4, Math::quaternion
- **NoSlotFound()**: Resources::PoolResourceMapper
- **Notify()**: FSM::Transition, Actions::Action, Actions::SequenceAction, Conditions::FSMCondition, Actions::Action, Actions::FSMAction, FSM::State, FSM::StateMachine
- NotifyEventHandlers() : Base::RenderDeviceBase, Base::DisplayDeviceBase, Base::RenderDeviceBase, Base::DisplayDeviceBase, Base::RenderDeviceBase
- NotifyGameLoad() : Game::GameServer
- NotifyGameSave() : Game::GameServer
- NotifyOfEntityTransformChange() :
  InternalGraphics::InternalStage
- nullvec() : Math::vector
- NumberOfPrimitives() : CoreGraphics::PrimitiveTopology
- NumberOfVertices() : CoreGraphics::PrimitiveTopology
- NumPools : Memory::PoolArrayAllocator
- NumSplits : Lighting::PSSMUtil
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **O -**
  - ObjectInspectorHandler() : Debug::ObjectInspectorHandler
  - ObjectRef() : Threading::ObjectRef
  - ObserverContext() : Visibility::ObserverContext
  - ObtainKeyboardCapture() : Base::InputServerBase
  - ObtainMouseCapture() : Base::InputServerBase
  - oldSlot : Resources::PoolLoadingResource
  - OnActivate() : GraphicsFeature::GraphicsProperty, Game::Manager, Game::Property, BaseGameFeature::TimeManager, BaseGameFeature::TimeSource, BaseGameFeature::CategoryManager, InternalGraphics::InternalGraphicsEntity, BaseGameFeature::EnvQueryManager, BaseGameFeature::GlobalAttrsManager, InternalGraphics::InternalGraphicsEntity, Game::Entity, InternalGraphics::InternalGraphicsEntity, Game::FeatureUnit, InternalGraphics::InternalGraphicsEntity, Game::Manager, Game::Property, GraphicsFeature::ChaseCameraProperty, Game::Property, GraphicsFeature::GraphicsProperty, GraphicsFeature::MayaCameraProperty, PhysicsFeature::ActorPhysicsProperty, PhysicsFeature::MouseGripperProperty, PhysicsFeature::PhysicsProperty,
OnAttach() : XInput::XInputGamePad, Base::KeyboardBase, Base::MouseBase, CoreGraphics::DisplayEventHandler, CoreGraphics::RenderEventHandler, CoreGraphics::DisplayEventHandler, Base::GamePadBase, Base::KeyboardBase, Base::MouseBase, XInput::XInputGamePad, Input::InputHandler, CoreGraphics::RenderEventHandler

OnAttachedToSequencer() : Animation::AnimJob

OnAttachEntity() : RenderModules::RTPluginRegistry, RenderModules::RTPlugin

OnAttachToModel() : Characters::CharacterNode, Models::ModelNode, Models::TransformNode, Models::ModelNode, Models::TransformNode, Particles::ParticleSystemNode

OnAttachToResource() : Resources::ResourceLoader, Resources::ResourceSaver, Resources::ResourceLoader, Resources::ResourceSaver

OnAttachToResourceManager() : Resources::ResourceMapper, Resources::SimpleResourceMapper, Resources::PoolResourceMapper

OnAttachToServer() : InternalGraphics::InternalStage, InternalGraphics::InternalView

OnAttachToStage() : InternalGraphics::InternalGraphicsEntity

OnAttachToView() : Graphics::CameraEntity, InternalGraphics::InternalCameraEntity

OnBeginFrame() : Game::Property, Game::Manager, Game::Property, XInput::XInputGamePad, Input::InputHandler, Base::KeyboardBase, Base::MouseBase, BaseGameFeature::EntityManager, Game::Entity, Game::FeatureUnit, Game::Manager,
Base::SkinnedMeshRendererBase, Game::Property, Game::FeatureUnit, Game::Manager, PhysicsFeature::MouseGripperProperty, Game::Manager, Game::FeatureUnit, PhysicsFeature::TriggerProperty, Game::Property, StateObjectFeature::StateProperty, Game::Manager, Game::FeatureUnit, Game::Manager, Base::KeyboardBase, Game::Manager, Base::MouseBase, Input::InputHandler, XInput::XInputGamePad, Base::SkinnedMeshRendererBase, Game::Property

- OnCancelRequest() : Resources::LoadingResource
- OnChar() : Win32::Win32DisplayDevice
- OnCloseRequested() : Win32::Win32DisplayDevice
- OnConfigureDisplay() : App::RenderApplication
- OnCopyFile() : IO::IoInterfaceHandler
- OnCreateDirectory() : IO::IoInterfaceHandler
- OnCreateManagedResource() : Resources::ResourceMapper, Resources::SimpleResourceMapper
- OnDeactivate() : GraphicsFeature::GraphicsProperty, Game::Manager, GraphicsFeature::CameraProperty, BaseGameFeature::TimeManager, BaseGameFeature::TimeSource, BaseGameFeature::EntityManager, BaseGameFeature::CategoryManager, BaseGameFeature::EnvQueryManager, InternalGraphics::InternalGraphicsEntity, BaseGameFeature::EnvEntityManager, Game::Entity, InternalGraphics::InternalGraphicsEntity, Game::FeatureUnit, Game::Manager, Game::Property, GraphicsFeature::CameraProperty, GraphicsFeature::GraphicsProperty, PhysicsFeature::ActorPhysicsProperty, GraphicsFeature::MayaCameraProperty, PhysicsFeature::EnvironmentCollideProperty, PhysicsFeature::MouseGripperProperty, PhysicsFeature::PhysicsProperty, PhysicsFeature::TriggerProperty, Game::Manager,
OnDeleteDirectory() : IO::IoInterfaceHandler
OnDeleteFile() : IO::IoInterfaceHandler
OnDiscardManagedResource() : Resources::ResourceMapper, Resources::SimpleResourceMapper
OnDiscardSharedData() :
   InternalGraphics::InternalGraphicsEntity,
   Graphics::GraphicsEntity,
   InternalGraphics::InternalGraphicsEntity,
   InternalGraphics::InternalModelEntity,
   Graphics::GraphicsEntity
OnDiscardStage() : RenderModules::RTPluginRegistry, RenderModules::RTPlugin
OnDiscardView() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry
OnEndFrame() : Game::Manager, Input::InputHandler, Game::Manager, Input::InputHandler, BaseGameFeature::EntityManager, Game::FeatureUnit, Game::Manager, Base::SkinnedMeshRendererBase, Game::FeatureUnit, Game::Manager, Game::Manager, Game::FeatureUnit, Game::Manager, Input::InputHandler, Base::SkinnedMeshRendererBase
OnEntry() : FSM::State
OnEvent() : Input::InputHandler, Base::KeyboardBase, Base::MouseBase, Base::KeyboardBase, Base::MouseBase, Input::InputHandler
OnExit() : FSM::State, Core::ExitHandler
OnFrame() : RenderModules::RenderModule, Http::HttpServer, Win32::Win32InputServer,
BaseGameFeature::TimeManager,
Animation::AnimEventHandlerBase,
BaseGameFeature::EnvQueryManager,
BaseGameFeature::FocusManager,
BaseGameFeature::GameStateHandler, Game::FeatureUnit,
Game::GameServer, Animation::AnimEventServer,
Game::Manager, GraphicsFeature::AttachmentManager,
Game::FeatureUnit, Game::Manager, FSM::State,
Game::Manager, Debug::DebugShapeRenderer,
Game::Manager, Debug::DebugTextRenderer,
Graphics::GraphicsServer, Game::Manager,
Base::InputServerBase, Game::Manager,
Win32::Win32InputServer,
InternalGraphics::AttachmentServer,
InternalGraphics::InternalGraphicsServer

- OnGainActivity(): Game::Property, Game::Entity, Game::Property
- OnHide(): Particles::ParticleSystemNodeInstance,
  InternalGraphics::InternalGraphicsEntity,
  Models::ModelNodeInstance,
  InternalGraphics::InternalGraphicsEntity,
  InternalGraphics::InternalModelEntity,
  Models::ModelInstance, Models::ModelNodeInstance
- OnKeyDown(): Win32::Win32DisplayDevice
- OnKeyUp(): Win32::Win32DisplayDevice
- OnKillFocus(): Win32::Win32DisplayDevice
- OnLoad(): Game::Property, Game::Manager,
  Game::Property, Game::FeatureUnit, Game::Property,
  BaseGameFeature::TimeManager,
  BaseGameFeature::TimeSource,
  BaseGameFeature::EntityManager, Game::Entity,
  Game::FeatureUnit, Game::Manager, Game::Property,
  Game::Manager, PhysicsFeature::TriggerProperty,
  Game::Manager, Game::FeatureUnit, Game::Property,
  Actions::SequenceAction, Script::DialogManager,
  StateObjectFeature::StateProperty, Game::Manager,
  Game::FeatureUnit, Game::Manager,
  BaseGameFeature::TimeSource, Game::Manager,
  BaseGameFeature::TimeSource, Game::Property
- **OnLoadCancelled()**: `Resources::ResourceLoader`, `Resources::StreamResourceLoader`, `Resources::ResourceLoader`, `Resources::StreamResourceLoader`, `Models::StreamModelLoader`

- **OnLoadRequested()**: `Resources::D3D9TextureStreamer`, `Resources::ResourceLoader`, `Resources::StreamResourceLoader`, `Resources::D3D9TextureStreamer`, `CoreGraphics::MemoryMeshLoader`, `Win360::D3D9MemoryIndexBufferLoader`, `Win360::D3D9MemoryVertexBufferLoader`, `Resources::StreamResourceLoader`, `Resources::ResourceLoader`, `Win360::D3D9MemoryIndexBufferLoader`, `Resources::ResourceLoader`, `Win360::D3D9MemoryVertexBufferLoader`, `Resources::StreamResourceLoader`, `Models::StreamModelLoader`

- **OnLoadResources()**: `StateObjectFeature::StateGraphicsProperty`

- **OnLoseActivity()**: `Game::Property`, `Game::Entity`, `Game::Property`, `PhysicsFeature::ActorPhysicsProperty`, `Game::Property`

- **OnLoseCameraFocus()**: `GraphicsFeature::CameraProperty`

- **OnLoseInputFocus()**: `GraphicsFeature::MayaCameraProperty`

- **OnLostDevice()**: `Direct3D9::D3D9ShaderInstance`

- **OnMinimized()**: `Win32::Win32DisplayDevice`

- **OnMountArchive()**: `IO::IoInterfaceHandler`

- **OnMouseButton()**: `Win32::Win32DisplayDevice`

- **OnMouseMove()**: `Win32::Win32DisplayDevice`

- **OnMouseWheel()**: `Win32::Win32DisplayDevice`

- **OnMoveAfter()**: `Game::Property`, `Game::Entity`, `Game::Property`, `PhysicsFeature::ActorPhysicsProperty`, `PhysicsFeature::MouseGripperProperty`, `PhysicsFeature::PhysicsProperty`, `Game::Property`

- **OnMoveBefore()**: `Game::Property`, `PhysicsFeature::ActorPhysicsProperty`, `Game::Entity`, `Game::Property`, `PhysicsFeature::MouseGripperProperty`, `PhysicsFeature::PhysicsProperty`, `Game::Property`
Game::Property

- OnNotifyCullingVisible():
  - InternalGraphics::InternalGraphicsEntity,
  - Models::ModelNodeInstance,
  - Characters::CharacterSkinNodeInstance,
  - Characters::CharacterNodeInstance,
  - Models::ModelNodeInstance,
  - InternalGraphics::InternalGraphicsEntity,
  - InternalGraphics::InternalModelEntity,
  - Models::ModelInstance, Models::ModelNodeInstance

- OnObtainCameraFocus(): GraphicsFeature::CameraProperty,
  GraphicsFeature::ChaseCameraProperty

- OnObtainCapture():
  - Input::InputHandler,
  - Base::KeyboardBase, Base::MouseBase,
  - Base::KeyboardBase, Base::MouseBase,
  - Input::InputHandler

- OnObtainInputFocus():
  - GraphicsFeature::MayaCameraProperty

- OnPaint(): Win32::Win32DisplayDevice

- OnPending():
  - Resources::ResourceLoader,
  - Resources::StreamResourceLoader,
  - Resources::ResourceLoader,
  - Resources::StreamResourceLoader,
  - Models::StreamModelLoader

- OnPrepare():
  - Resources::SimpleResourceMapper,
  - Resources::PoolResourceMapper,
  - Resources::ResourceMapper

- OnProcessInput():
  - App::RenderApplication,
  - App::ViewerApplication

- OnReadStream(): IO::IoInterfaceHandler

- OnRegister():
  - RenderModules::RTPlugin

- OnReleaseCapture():
  - Input::InputHandler,
  - Base::KeyboardBase, Base::MouseBase,
  - Base::KeyboardBase, Base::MouseBase,
  - Input::InputHandler

- OnRemove():
  - Input::InputHandler,
  - CoreGraphics::DisplayEventHandler, Input::InputHandler,
  - CoreGraphics::DisplayEventHandler,
  - CoreGraphics::RenderEventHandler, Input::InputHandler,
CoreGraphics::RenderEventHandler
- OnRemoveEntity() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry
- OnRemoveFromMapper() : Resources::TexturePoolMapperScheduler, Resources::ResourceScheduler
- OnRemoveFromModel() : Characters::CharacterNode, Models::ModelNode, Particles::ParticleSystemNode
- OnRemoveFromResource() : Resources::ResourceLoader, Resources::ResourceSaver, Resources::ResourceLoader, Resources::ResourceSaver
- OnRemoveFromResourceManager() : Resources::PoolResourceMapper, Resources::ResourceMapper, Resources::SimpleResourceMapper
- OnRemoveFromSequencer() : Animation::AnimJob
- OnRemoveFromServer() : InternalGraphics::InternalStage, InternalGraphics::InternalView
- OnRemoveFromStage() : InternalGraphics::InternalGraphicsEntity
- OnRemoveFromView() : Graphics::CameraEntity, InternalGraphics::InternalCameraEntity
- OnRender() : Game::Property, Game::Entity, GraphicsFeature::CameraProperty, Game::Property, GraphicsFeature::ChaseCameraProperty, Game::Property, GraphicsFeature::CameraProperty, GraphicsFeature::MayaCameraProperty, Game::Property
- OnRenderAfter() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry
- **OnRenderDebug()**: `GraphicsFeature::GraphicsProperty`, `Game::Property`, `GraphicsFeature::GraphicsProperty`, `Game::Property`, `Visibility::VisibilitySystemBase`, `Game::Manager`, `Game::Property`, `PhysicsFeature::TriggerProperty`, `Game::Property`, `Visibility::VisibilityChecker`, `Visibility::VisibilityBoxSystem`, `Visibility::VisibilityQuadtree`, `Visibility::VisibilityClusterSystem`, `BaseGameFeature::TimeManager`, `Game::Property`, `GraphicsFeature::GraphicsProperty`, `BaseGameFeature::EntityManager`, `Game::Property`, `Game::Entity`, `Game::FeatureUnit`, `PhysicsFeature::MouseGripperProperty`, `PhysicsFeature::ActorPhysicsProperty`, `Game::Property`, `Game::Manager`, `Game::FeatureUnit`, `GraphicsFeature::GraphicsProperty`, `Game::Manager`, `Game::Property`, `Game::Manager`, `Game::FeatureUnit`, `InternalGraphics::InternalGraphicsEntity`, `Game::Manager`, `Game::Property`, `Game::Manager`, `Game::Property`, `InternalGraphics::InternalGraphicsEntity`, `InternalGraphics::InternalModelEntity`, `InternalGraphics::InternalStage`, `Game::Property`, `Lighting::InternalAbstractLightEntity`, `Game::Manager`, `Lighting::InternalGlobalLightEntity`, `Lighting::InternalPointLightEntity`, `Lighting::InternalSpotLightEntity`

- **OnRenderFrameBatch()**: `RenderModules::RTPlugin`, `RenderModules::RTPluginRegistry`

- **OnRender WithoutView()**: `RenderModules::RTPlugin`, `RenderModules::RTPluginRegistry`

- **OnRequestManagedResource()**: `Resources::TexturePoolMapperScheduler`, `Resources::ResourceScheduler`

- **OnRequestOtherMipMap()**: `Resources::TexturePoolMapperScheduler`

- **OnReset()**: `Base::KeyboardBase`, `Base::MouseBase`, `Base::GamePadBase`, `Base::KeyboardBase`, `Base::MouseBase`, `Input::InputHandler`

- **OnResetDevice()**: `Direct3D9::D3D9ShaderInstance`
OnResetSharedData():
  InternalGraphics::InternalGraphicsEntity,
  InternalGraphics::InternalModelEntity
OnResolveVisibility():
  Lighting::InternalAbstractLightEntity,
  InternalGraphics::InternalGraphicsEntity,
  Lighting::InternalAbstractLightEntity,
  InternalGraphics::InternalGraphicsEntity,
  InternalGraphics::InternalModelEntity,
  Lighting::InternalAbstractLightEntity
OnResourcesLoaded():
  Characters::CharacterNode,
  Models::ModelNode,
  Particles::ParticleSystemNode,
  Models::ModelNode
OnRestored():
  Win32::Win32DisplayDevice
OnSave():
  Game::Property,
  BaseGameFeature::TimeSource,
  Game::Manager,
  Game::Property,
  Resources::ResourceSaver,
  Game::Property,
  Game::Manager,
  Game::Property,
  Game::Manager,
  Game::FeatureUnit,
  Game::Manager,
  Base::StreamTextureSaverBase,
  Game::Manager,
  BaseGameFeature::EntityManager,
  Script::DialogManager,
  PhysicsFeature::TriggerProperty,
  StateObjectFeature::StateProperty,
  Game::Manager,
  Game::Entity,
  Win360::D3D9StreamTextureSaver,
  Game::FeatureUnit,
  Win360::D3D9StreamTextureSaver,
  Game::Manager,
  Game::Property,
  Game::Manager,
  Game::Property,
  Game::FeatureUnit,
  Game::Manager,
  BaseGameFeature::TimeSource,
  Game::Manager,
  BaseGameFeature::TimeSource,
  Game::Manager,
  Game::Property,
  Game::FeatureUnit,
  Game::Property,
  BaseGameFeature::TimeManager,
  Game::Property
OnSetCursor():
  Win32::Win32DisplayDevice
OnSetFocus():
  Win32::Win32DisplayDevice
OnSetOverwriteColor():
  GraphicsFeature::GraphicsProperty
OnSetShaderVariable():
  GraphicsFeature::GraphicsProperty
OnSetupResourceMappers():
  App::RenderApplication
OnSetupSharedData():
  InternalGraphics::InternalGraphicsEntity,
  Graphics::GraphicsEntity,
  InternalGraphics::InternalGraphicsEntity,
InternalGraphics::InternalModelEntity, Graphics::GraphicsEntity
- OnStageCreated() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry
- OnStart() : Game::Property, Game::Entity, Game::Property, Game::FeatureUnit, GraphicsFeature::CameraProperty, Game::Property, Game::FeatureUnit, Game::Property, GraphicsFeature::CameraProperty, Game::Manager, Game::FeatureUnit, Game::Property, GraphicsFeature::CameraProperty, Game::Property, Game::Manager, BaseGameFeature::EntityManager, Game::Manager, Game::Manager, Game::Property
- OnStateEnter() : BaseGameFeature::GameStateHandler
- OnStateLeave() : BaseGameFeature::GameStateHandler
- OnSuccessRequest() : Resources::PoolLoadingResource, Resources::LoadingResource
- OnSwitchActiveState() : StateObjectFeature::StateGraphicsProperty
- OnToggleFullscreenWindowed() : Win32::Win32DisplayDevice
- OnUnregister() : RenderModules::RTPlugin
- OnUpdate() : Resources::SimpleResourceManager
Resources::ResourceMapper
- OnUpdateAfter() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry
- OnUpdateBefore() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry
- OnUpdateFrame() : App::ViewerApplication, App::RenderApplication
- OnViewCreated() : RenderModules::RTPluginRegistry, RenderModules::RTPlugin
- OnVisibilityResolve() : Models::ModelNodeInstance, Particles::ParticleSystemNodeInstance, Models::ModelNodeInstance, Models::ShapeNodeInstance, Models::ModelInstance, Models::ModelNodeInstance, Models::ShapeNodeInstance
- OnWriteStream() : IO::IoInterfaceHandler
- Open() : Core::CoreServer, IO::Stream, IO::StreamReader, Http::HttpMessageHandler, Direct3D9::D3D9TextRenderer, BaseGameFeature::LoaderServer, Lighting::SM30LightServer, InternalGraphics::InternalGraphicsServer, Http::SvgLineChartWriter, Messaging::AsyncPort, Resources::ResourceManager, Graphics::Display, App::ViewerApplication, Win32::Win32InputServer, IO::StreamReader, Interface::InterfaceBase, Base::TextRendererBase, Win32::Win32InputServer, App::GameApplication, FSM::StateMachine, Base::DisplayDeviceBase, Direct3D9::D3D9TextRenderer, Visibility::VisibilityQuadtree, App::ConsoleApplication, Base::ShapeRendererBase, IO::XmlWriter, Visibility::VisibilitySystemBase, IO::ConsoleHandler, IO::BXmlReader, IO::ConsoleHandler, IO::StreamWriter, IO::Console, Lighting::SM30ShadowServer, Game::GameServer, IO::ConsoleHandler, App::RenderApplication, Graphics::GraphicsInterface, Messaging::Handler, Net::StdTcpServer, Http::SvgPageWriter, Graphics::GraphicsHandler, Win32::Win32DisplayDevice, Base::VertexLayoutServerBase, IO::ZipFileStream, Http::HttpServerProxy, Direct3D9::D3D9ShaderServer, IO::StreamWriter,
IO::StreamReader, IO::StreamWriter,
Visibility::VisibilityBoxSystem, Debug::DebugHandler,
IO::ZipFileEntry, IO::BinaryReader, IO::StreamReader,
Http::HtmlPageWriter, Base::VertexLayoutServerBase,
IO::IInterfaceHandler, Direct3D9::D3D9RenderDevice,
Animation::AnimEventServer, Messaging::Handler,
Base::TransformDeviceBase, App::Application,
IO::StreamReader, Visibility::VisibilityChecker,
IO::StreamWriter, Debug::DebugInterface,
Direct3D9::D3D9ShaderServer, Models::VisResolver,
Win360::D3D9TransformDevice, IO::StreamWriter,
IO::FileStream, Visibility::VisibilityClusterSystem,
IO::XmlReader, Graphics::GraphicsServer,
Win360::Win360Socket, Base::InputServerBase,
Particles::ParticleServer, Base::ShaderServerBase,
Net::StdTcpServer, Models::ModelServer,
Win32::Win32DisplayDevice, Lighting::LightPrePassServer,
Win32::Win32DisplayDevice, Http::HttpInterface,
IO::MemoryStream, InternalGraphics::AttachmentServer,
IO::BinaryWriter, Base::RenderDeviceBase,
Http::HttpServer, Frame::FrameServer,
IO::LogFileConsoleHandler, Direct3D9::D3D9RenderDevice,
Win360::Win360Socket, Win360::D3D9ShapeRenderer,
Lighting::LightServerBase, Lighting::LightPrePassServer,
IO::ConsoleHandler, IO::ExcelXmlReader,
Win360::D3D9ShapeRenderer,
Win360::D3D9TransformDevice, Messaging::Handler,
Lighting::SM30ShadowServer
- OpenDInputMouse() : Win32::Win32InputServer
- OpenFile() : OSX::OSXFSWrapper,
  Win360::Win360FSWrapper
- OpenProgressIndicator() : BaseGameFeature::LoaderServer
- OpenWindow() : Win32::Win32DisplayDevice
- Operator() : Conditions::Operator<TYPE >
- operator *() : Math::float2, Util::List<TYPE >::Iterator,
  Math::vector, WeakPtr<TYPE >, Math::float4, Ptr<TYPE >,
  Math::float4
- operator *=() : Math::float4, Math::vector, Math::float2,
  Math::float4
operator bool() : Util::List< TYPE >::Iterator
operator delete() : Win32::Win32Guid, Util::String, OSX::OSXGuid, Util::Blob
operator new() : Util::String, Util::Blob, Win32::Win32Guid, OSX::OSXGuid
operator TYPE *() : WeakPtr< TYPE >, Ptr< TYPE >
operator!=(-) : Util::FourCC, Util::StringAtomTableBase::StaticString, Util::Stack< TYPE >, Util::FixedTable< TYPE >, Math::float4, IO::URI, Util::Variant, Util::FixedArray< TYPE >, Util::Variant, Win32::Win32Guid, Util::Variant, Ptr< TYPE >, Util::KeyValuePair< KEYTYPE, VALUETYPE >, Util::Blob, Animation::AnimEventInfo, Math::vector, CoreAnimation::AnimEvent, Math::float4, Win360::Win360FileTime, IO::MediaType, Util::Variant, Util::StringAtom, Util::Array< TYPE >, Util::Queue< TYPE >, Util::Variant, Util::StringAtom, Core::Rtti, Util::BitField< NUMBITS >, CoreGraphics::DisplayMode, Util::Array< TYPE >, Math::float2, Math::quaternion, Math::float4, Ptr< TYPE >, Math::matrix44, OSX::OSXGuid, Util::Queue< TYPE >, Math::point, Util::Variant, Util::List< TYPE >::Iterator, Util::String, Util::StringAtom, Util::Variant, OSX::OSXFileTime
operator()() : Util::Delegate< ARGTYPE >
operator+() : Math::float4, Math::point, Math::float4, Math::float2, Math::vector
operator++() : Util::List< TYPE >::Iterator
operator+=() : Math::vector, Math::point, Math::float2, Math::float4, Util::String
operator-() : Math::float4, Math::float2, Math::float4, Math::float2, Math::vector, Math::point, Math::float4, Math::point
operator--() : Util::List< TYPE >::Iterator
operator-() : Math::float4, Math::vector, Math::float4, Math::point, Math::float2
operator->() : Ptr< TYPE >, Util::List< TYPE >::Iterator, WeakPtr< TYPE >
operator< : CoreAnimation::AnimEvent, Util::StringAtom, Resources::LoadingResource, Win360::Win360IpAddress,
Win32::Win32Guid, OSX::OSXFileTime, Util::KeyValuePair<KEYTYPE, VALUETYPE>, Util::StringAtomTableBase::StaticString, Win360::Win360FileTime, Util::Blob, Util::FourCC, OSX::OSXGuid, Animation::AnimEventInfo, Util::Variant, Util::String

- operator<=: Util::String, Animation::AnimEventInfo, Util::Variant, OSX::OSXGuid, Util::Blob, Util::StringAtom, Win32::Win32Guid, Util::Variant, Ptr<TYPE>, Math::quaternion, Util::Variant, Util::Array<TYPE>, Util::FixedSizeTable<TYPE>, Math::point, Util::Variant, OSX::OSXGuid, Util::StringAtom, Math::float4, Util::Variant, Util::RingBuffer<TYPE>, Util::StringAtom, Threading::SafeQueue<TYPE>, Util::Variant, Math::float2, IO::URI, Util::KeyValuePair<KEYTYPE, VALUETYPE>, Math::matrix44, Util::Variant, Ptr<TYPE>, Math::float4, Util::Blob, Util::Variant, Math::polar, Util::Variant, IO::MediaType, Util::FixedArray<TYPE>, WeakPtr<TYPE>, Util::List<TYPE>, Win32::Win32Guid, Math::vector, Threading::SafePriorityQueue<PRITYPE, TYPE>, Util::Queue<TYPE>, Math::vector, OSX::OSXGuid, Util::String, Util::Variant, Util::Dictionary<KEYTYPE, VALUETYPE>, Math::point, Math::quaternion, Util::BitField<NUMBITS>, Util::StringAtom, Util::Variant, Util::String, Util::Stack<TYPE>

- operator==(): Util::StringAtom, Win32::Win32Guid, Util::KeyValuePair<KEYTYPE, VALUETYPE>, OSX::OSXFileTime, Util::StringAtom, Messaging::Id, Util::Variant, Math::float2, Util::Queue<TYPE>, Util::Variant, Win360::Win360FileTime, Util::Variant, IO::MediaType, Math::quaternion, Math::point, Util::String, Math::float4, CoreGraphics::DisplayMode, Math::vector, Ptr<TYPE>, Util::Variant, Math::float4, Util::Queue<TYPE>, Util::Variant, Ptr<TYPE>, CoreAnimation::AnimEvent, OSX::OSXGuid,
Util::BitField< NUMBITS >, Core::Rtti, IO::URI, Util::Variant, Util::FourCC, Math::matrix44, Util::Blob, Util::String, Math::float4, Util::String, Util::FixedArray< TYPE >, Util::List< TYPE >::Iterator, Animation::AnimEventInfo, Util::Stack< TYPE >, Win360::Win360IpAddress, Util::Variant, Util::FixedTable< TYPE >, Util::StringAtomTableBase::StaticString, Util::Variant, Util::Array< TYPE >

- operator(): Win360::Win360IpAddress, Util::StringAtom, Util::Variant, Util::KeyValuePair< KEYTYPE, VALUETYPE >, Win32::Win32Guid, Util::FourCC, CoreAnimation::AnimEvent, Util::Blob, Animation::AnimEventInfo, Util::String, OSX::OSXFileTime, Win360::Win360FileTime, OSX::OSXGuid, Util::StringAtomTableBase::StaticString
- operator>=: Animation::AnimEventInfo, Util::Blob, Util::KeyValuePair< KEYTYPE, VALUETYPE >, Util::String, Util::FourCC, Util::Variant, Win32::Win32Guid, Util::StringAtom, OSX::OSXGuid, CoreAnimation::AnimEvent
- operator[][](): Util::Queue< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::Stack< TYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::Queue< TYPE >, Util::FixedArray< TYPE >, Util::String, Util::Array< TYPE >, Util::String, Util::SimpleTree< VALUETYPE >::Node, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::RingBuffer< TYPE >, Util::Array< TYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >
- Or(): Util::BitField< NUMBITS >
- origin(): Math::point
- ortholh(): Math::matrix44
- orthooffcenterlh(): Math::matrix44
- orthooffcenterrh(): Math::matrix44
- orthorh(): Math::matrix44
- OSXCriticalSection(): OSX::OSXCriticalSection
- OSXFileTime(): OSX::OSXFileTime
- OSXGuid(): OSX::OSXGuid
- OSXHeap(): OSX::OSXHeap
- OSXMemoryPool(): OSX::OSXMemoryPool
- OSXThread() : **OSX::OSXThread**
- OSXThreadLocalPtr() : **OSX::OSXThreadLocalPtr< TYPE >**
- OverwriteScriptManager() : **ScriptFeature::ScriptFeatureUnit**
- Owner() : **FrameSync::FrameSyncSharedData**
- OwnerDiscard() : **FrameSync::FrameSyncSharedData**
- OwnerSetup() : **FrameSync::FrameSyncSharedData**
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **p** -
  
  - PacketSize : Net::DebugPacket
  - Parent() : Util::SimpleTree< VALUETYPE >::Node
  - ParseArgs() : Actions::Action, Conditions::Condition, Actions::Action, Conditions::Condition
  - ParseBluePrints() : BaseGameFeature::FactoryManager
  - ParseDataTag() : Characters::CharacterNode, Characters::CharacterSkinNode, Models::ShapeNode, Models::StateNode, Models::TransformNode, Models::ModelNode, Particles::ParticleSystemNode
  - ParseFile() : StateObjectFeature::StateProperty
  - ParseKeyValuePairs() : Util::String
  - ParseQuery() : IO::URI
  - ParticleRenderer() : Particles::ParticleRenderer
  - ParticleRendererBase() : Base::ParticleRendererBase
  - ParticleServer() : Particles::ParticleServer
  - ParticleSystem() : Particles::ParticleSystem
  - ParticleSystemNode() : Particles::ParticleSystemNode
  - ParticleSystemNodeInstance() : Particles::ParticleSystemNodeInstance
  - PatchInputDesc() : Base::JobBase
  - PatchOutputDesc() : Base::JobBase
  - PatchUniformDesc() : Base::JobBase
  - Pause() : Debug::DebugTimer
BaseGameFeature::TimeSource
- PauseAll() : BaseGameFeature::TimeManager
- Peek() : Threading::SafePriorityQueue< PRITYPE, TYPE >, Messaging::AsyncPort, Win360::Win360Event, Threading::SafeQueue< TYPE >, Util::Queue< TYPE >, Messaging::AsyncPort, Util::Stack< TYPE >, Messaging::AsyncPort, Win360::Win360Event
- PerformVisibilityQuery() : Visibility::VisibilityChecker
- permute() : Math::float4
- permute_control() : Math::float4
- perspfovlh() : Math::matrix44
- perspfovrh() : Math::matrix44
- persplh() : Math::matrix44
- perspoffcenterlh() : Math::matrix44
- perspoffcenterrh() : Math::matrix44
- persprh() : Math::matrix44
- PhysicsProperty() : PhysicsFeature::PhysicsProperty
- plane() : Math::plane
- Platform : Base::SystemInfoBase
- PlatformAsString() : Base::SystemInfoBase
- PlayClipJob() : Animation::PlayClipJob
- PlayEmotes() : Script::DialogManager
- PlaySound() : Script::DialogManager
- point() : Math::point
- pointat() : Math::line
- PointLightEntity() : Graphics::PointLightEntity
- polar() : Math::polar
- Polygon() : Http::SvgPageWriter
- PolyLine() : Http::SvgPageWriter
- PoolArrayAllocator() : Memory::PoolArrayAllocator
- PoolLoadingResource() : Resources::PoolLoadingResource
- poolMapper : Resources::TexturePoolMapperScheduler
- PoolResourceMapper() : Resources::PoolResourceMapper
- Pop() : Util::Stack< TYPE >
- Port() : IO::URI
- Position : IO::Stream
- PrecomputeKeySliceValues() : CoreAnimation::AnimClip
- PreloadDialogTables() : Script::DialogManager
- PreloadTextures() : Graphics::MouseRenderer,
Prepared

PreSample()

Present()

print()

Print()

print()

Print()

print()

print()

print()

Priority

ProcessWindowMessages()

project_screen_rh()

Property()

Protocol

PSSMUtil()

Ptr()

Push()

PushFlush()

PushJob()

PushJobChain()

PushJobCommand()

PushJobSlices()

PushSync()
- PutBool() : `Script::ActionReader`
- PutClass() : `Script::ActionReader`
- PutEntity() : `Script::ActionReader`
- PutEvent() : `CoreGraphics::DisplayEventHandler`, `CoreGraphics::ThreadSafeDisplayEventHandler`, `Base::InputServerBase`, `CoreGraphics::ThreadSafeDisplayEventHandler`, `CoreGraphics::RenderEventHandler`, `Base::InputServerBase`, `CoreGraphics::ThreadSafeRenderEventHandler`
- PutFloat() : `Script::ActionReader`
- PutFloat4() : `Script::ActionReader`
- PutInt() : `Script::ActionReader`
- PutRequest() : `Http::HttpRequestHandler`
- PutString() : `Script::ActionReader`
- PutVector() : `Script::ActionReader`
- PutVersion() : `Script::ActionReader`
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- q -

- quaternion() : Math::quaternion
- Query() : IO::URI
- Queue() : Util::Queue< TYPE >
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
 a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **r** -
  - Rand() : Util::RandomNumberTable
  - Read() : IO::MemoryStream, OSX::OSXFSWrapper, IO::Stream, System::Win32Environment, IO::Stream, IO::FileStream, Conditions::Condition, Win360::Win360FSWrapper, IO::ZipFileEntry, IO::ZipFileStream, Actions::Action, Actions::ActionList, Actions::IfThenElseAction, Actions::SequenceAction, Conditions::And, Conditions::Condition, Conditions::Not, Conditions::Or, Actions::Action
  - ReadAll() : IO::TextReader
  - ReadAllLines() : IO::TextReader
  - ReadBlob() : IO::BinaryReader
  - ReadBool() : IO::BinaryReader
  - ReadChar() : IO::TextReader, IO::BinaryReader
  - ReadDInputMouse() : Win32::Win32InputServer
  - ReadDouble() : IO::BinaryReader
  - ReadFloat() : IO::BinaryReader
  - ReadFloat2() : IO::BinaryReader
  - ReadFloat4() : IO::BinaryReader
  - ReadFloatFromNormalizedUByte2() : IO::BinaryReader
  - ReadFloatFromUnsignedNormalizedUByte2() : IO::BinaryReader
  - ReadGuid() : IO::BinaryReader
  - ReadGuid() : IO::BinaryReader
- ReadInt() : System::Win32Registry, IO::BinaryReader
- ReadLine() : IO::TextReader
- ReadMatrix44() : IO::BinaryReader
- ReadMessage() : Messaging::MessageReader
- ReadPoint() : IO::BinaryReader
- ReadRawData() : IO::BinaryReader
- ReadRequest() : Http::HttpRequestReader
- ReadResponse() : Http::HttpResponseReader
- ReadShort() : IO::BinaryReader
- ReadString() : IO::BinaryReader, System::Win32Registry
- ReadTexturePoolFromXML() : Resources::PoolResourceMapper
- ReadUChar() : IO::BinaryReader
- ReadUInt() : IO::BinaryReader
- ReadUShort() : IO::BinaryReader
- ReadValue() : Math::Extrapolator< TYPE >
- ReadVector() : IO::BinaryReader
- Realloc() : OSX::OSXHeap, Win360::Win360Heap, Util::Array< TYPE >
- reciprocal() : Math::float4
- Rect() : Http::SvgPageWriter
- rectangle() : Math::rectangle< TYPE >
- RecurseCreateNodeInstanceHierarchy() : Models::ModelNode
- Recv() : Net::MessageClient, Net::MessageClientConnection, Net::StdTcpClient, Net::StdTcpClientConnection, Net::StdTcpServer, Win360::Win360Socket, Net::StdTcpClient, Net::StdTcpClientConnection, Win360::Win360Socket, Net::StdTcpServer
- RecvFrom() : Win360::Win360Socket
- Ref() : Threading::ObjectRef
- RefCounted() : Core::RefCounted
- reflect() : Math::float4, Math::matrix44
- Register() : Core::Factory
- RegisterAnimEventHandler() : Animation::AnimEventServer
- RegisterDebugCounter() : Debug::DebugServer
- RegisterDebugTimer() : Debug::DebugServer
- RegisterEntity() : Visibility::VisibilityChecker
- RegisterMessage() : Messaging::Port
- RegisterPropertyCallback() : Game::Entity
- RegisterRenderModule() : Graphics::GraphicsServer
- RegisterRTPlugin() : RenderModules::RTPluginRegistry
- RegisterSoftwareSkinnedMesh() : Base::SkinnedMeshRendererBase
- RegisterUnmanagedResource() : Resources::ResourceManager
- RegisterUriScheme() : IO::SchemeRegistry
- Release() : Core::RefCounted, Util::SimpleTree< VALUETYPE >::Node, Core::RefCounted
- ReleaseKeyboardCapture() : Base::InputServerBase
- ReleaseMouseCapture() : Base::InputServerBase
- ReleaseResources() : Resources::ResourceManager
- ReloadScripts() : Script::ScriptManager
- Remove() : Util::List< TYPE >
- RemoveAllEntities() : InternalGraphics::InternalStage
- RemoveAllHandlers() : Messaging::Port
- RemoveAllLoaders() : BaseGameFeature::LoaderServer
- RemoveAllMappers() : Resources::ResourceManager
- RemoveBack() : Util::List< TYPE >
- RemoveDisplayEventHandler() : Graphics::Display
- RemoveEntity() : InternalGraphics::InternalStage, BaseGameFeature::EntityManager
- RemoveEntityFromTriggered() : BaseGameFeature::EntityManager
- RemoveEntityImmediate() : BaseGameFeature::EntityManager
- RemoveEntityLoader() : BaseGameFeature::LoaderServer
- RemoveEventHandler() : Base::DisplayDeviceBase, Base::RenderDeviceBase
- RemoveFront() : Util::List< TYPE >
- RemoveGameFeature() : Game::GameServer
- RemoveHandler() : Messaging::AsyncPort, Messaging::Port, Messaging::HandlerThreadBase, Messaging::Port, Messaging::HandlerThreadBase, Messaging::Port, Messaging::AsyncPort, Messaging::Port, Messaging::AsyncPort, Messaging::Port, Messaging::AsyncPort, Messaging::Port, IO::Console, Messaging::HandlerThreadBase, Messaging::Port, Messaging::HandlerThreadBase, Messaging::Port
- RemoveInputHandler() : Base::InputServerBase
- RemoveInvalidAttachments() : \n  InternalGraphics::AttachmentServer
- RemoveManager() : Game::FeatureUnit
- RemoveMapper() : Resources::ResourceManager
- RemoveNode() : Models::Model
- RemoveNodeInstance() : Models::ModellInstance
- RemoveNullEntriesFromArray() : \n  BaseGameFeature::EntityManager
- RemovePort() : Messaging::Dispatcher
- RemoveRenderEventHandler() : Graphics::Display
- RemoveRequestHandler() : Http::HttpServer, Http::HttpServerProxy
- RemoveSkin() : Characters::CharacterSkinSet
- RemoveTimeSource() : BaseGameFeature::TimeManager
- RemoveVisibilityContext() : Visibility::VisibilityClusterSystem, Visibility::VisibilityQuadtree, Visibility::VisibilitySystemBase, Visibility::VisibilityBoxSystem
- RemoveVisibilitySystem() : Visibility::VisibilityChecker
- RenameLevel() : BaseGameFeature::CategoryManager
- Render() : Models::ModelNodeInstance, Direct3D9::D3D9ParticleSystemInstance, Models::ShapeNodeInstance, Models::ModelNodeInstance, InternalGraphics::InternalView, Frame::FrameBatch, Frame::FramePass, Particles::ParticleSystemNodeInstance, Models::ModelNodeInstance, Frame::FramePostEffect, Frame::FramePassBase, Direct3D9::D3D9ParticleSystemInstance, Characters::CharacterSkinNodeInstance, Models::ModelNodeInstance, Frame::FrameShader
- RenderApplication() : App::RenderApplication
- RenderDebug() : Models::TransformNodeInstance, Particles::ParticleSystemNodeInstance, Models::ModellInstance, InternalGraphics::InternalView, Models::ModelNodeInstance, Models::TransformNodeInstance, Models::ModelNodeInstance, Models::ModelNodeInstance, Characters::CharacterSkeletonInstance, Characters::CharacterInstance, Models::TransformNodeInstance,
Characters::CharacterNodeInstance
- RenderDebugVisualization() : PhysicsFeature::TriggerProperty
- RenderDevice() : CoreGraphics::RenderDevice
- RenderDeviceBase() : Base::RenderDeviceBase
- RenderEvent() : CoreGraphics::RenderEvent
- RenderEventHandler() : CoreGraphics::RenderEventHandler
- RenderGlobalLight() : Lighting::LightPrePassServer
- RenderLights() : Lighting::LightServerBase, Lighting::LightPrePassServer, Lighting::LightServerBase
- RenderModule() : RenderModules::RenderModule
- RenderParticleSystem() : Base::ParticleRendererBase
- RenderPointers() : Base::MouseRenderDeviceBase
- RenderPointLights() : Lighting::LightPrePassServer
- RenderShape() : CoreGraphics::RenderShape
- RenderSpotLights() : Lighting::LightPrePassServer
- RenderTargetBase() : Base::RenderTargetBase
- ReplaceChars() : Util::String
- ReplaceIllegalFilenameChars() : Util::String
- RequestResourceForLoading() : Resources::ResourceManager
- RequestState() : App::GameApplication
- Reserve() : Util::Blob, Util::Array< TYPE >, Util::String, Util::Array< TYPE >, Util::Queue< TYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::Queue< TYPE >
- ReserveFragments() : Characters::CharacterSkinNode
- ReserveSkinLists() : Characters::CharacterSkinLibrary
- ReserveSkins() : Characters::CharacterSkinLibrary
- Reset() : Base::InputServerBase, Resources::ResourceLoader, Resources::D3D9TextureStreamer, Base::InputServerBase, BaseGameFeature::TimeSource, Math::Extrapolator< TYPE >, Resources::ResourceLoader, Util::Array< TYPE >, Resources::PoolResourceMapper, Win360::Win360Event, Resources::ResourceLoader, Models::VisResolveContainer< TYPE >, Resources::ResourceLoader, Util::RingBuffer< TYPE >, Resources::ResourceLoader, BaseGameFeature::TimeSource, Resources::ResourceLoader, BaseGameFeature::TimeSource, Resources::ResourceLoader, Debug::DebugCounter,
Util::Array< TYPE >, Win360::Win360Event, Resources::ResourceLoader, Win360::Win360Timer, BaseGameFeature::TimeSource, Resources::ResourceLoader, Math::Extrapolator< TYPE >, Resources::ResourceLoader, RenderUtil::MayaCameraUtil, Resources::D3D9TextureStreamer, Resources::ResourceLoader, Base::InputServerBase

- ResetAccum() : Debug::DebugTimer
- ResetAll() : BaseGameFeature::TimeManager
- ResetFeatureBits() : Base::ShaderServerBase
- ResetModelNodeInstanceIndex() : Models::ModelServer
- ResetScreenSpaceStats() : Models::ModelNode
- ResetTime() : FrameSync::FrameSyncTimer
- Resize() : Util::FixedArray< TYPE >
- ResolveAssigns() : IO::AssignRegistry
- ResolveAssignsInString() : IO::AssignRegistry
- ResolveDepthBuffer() : Base::RenderTargetBase
- ResolveVisibleLights() : InternalGraphics::InternalView
- ResolveVisibleModelNodeInstances() : InternalGraphics::InternalView
- Resource() : Resources::Resource
- ResourceBase() : Base::ResourceBase
- ResourceDictionary() : Resources::ResourceDictionary
- ResourceLoader() : Resources::ResourceLoader
- ResourceManager() : Resources::ResourceManager
- ResourceMapper() : Resources::ResourceMapper
- ResourceSaver() : Resources::ResourceSaver
- resourceStreamingLevelOfDetail : Models::ModelNode
- ReuseMips() : Resources::D3D9TextureStreamer
- RingBuffer() : Util::RingBuffer< TYPE >
- Root() : Util::SimpleTree< VALUETYPE >
- rotationaxis() : Math::quaternion, Math::matrix44
- rotationmatrix() : Math::matrix44, Math::quaternion
- rotationquaternion() : Math::matrix44
- rotationx() : Math::matrix44
- rotationy() : Math::matrix44
- rotationyawpitchroll() : Math::quaternion, Math::matrix44
- rotationz() : Math::matrix44
- Row() : Util::QuadTree< TYPE >::Node
- RowIndex() : **BaseGameFeature::CategoryManager::Entry**
- RTPlugin() : **RenderModules::RTPlugin**
- RTPluginRegistry() : **RenderModules::RTPluginRegistry**
- Rtti() : **Core::Rtti**
- Run() : **App::GameApplication**, **App::Application**, **App::RenderApplication**, **App::Application**, **Visibility::VisibilityQuery**
- Running() : **Win360::Win360Timer**
- RunThroughHandlerThread() : **Messaging::RunThroughHandlerThread**
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **S** -

- SafeFlag() : Threading::SafeFlag
- SafePriorityQueue() : Threading::SafePriorityQueue< PRITYPE, TYPE >
- SafeQueue() : Threading::SafeQueue< TYPE >
- Sample() : CoreAnimation::AnimUtil, Particles::EnvelopeCurve
- Save() : Script::Task, Resources::Resource, BaseGameFeature::UserProfile
- SaveAttributes() : BaseGameFeature::GlobalAttrsManager
- SaveDialogs() : Script::DialogManager
- SaveDialogTakeStates() : Script::DialogManager
- SaveScreenshot() : Direct3D9::D3D9RenderDevice, Base::RenderDeviceBase
- scale() : Math::matrix44
- scaling() : Math::matrix44
- Scheme() : IO::URI
- SchemeRegistry() : IO::SchemeRegistry
- ScriptFeatureUnit() : ScriptFeature::ScriptFeatureUnit
- ScriptManager() : Script::ScriptManager
- Seek() : IO::Stream, IO::FileStream, OSX::OSXFSWrapper, IO::MemoryStream, IO::Stream, Win360::Win360FSWrapper, IO::ZipFileStream
- SeekOrigin : IO::Stream
- SegmentedGfxUtil() : GraphicsFeature::SegmentedGfxUtil
- SelectActiveVariation() : Base::ShaderInstanceBase, Direct3D9::D3D9ShaderInstance
- Semantic : Base::ShaderVariableBase
- Send() : Win360::Win360Socket, Messaging::AsyncPort, Net::StdTcpClient, Net::StdTcpClientConnection, Messaging::Port, Win360::Win360Socket, Messaging::Port, Messaging::AsyncPort, Messaging::Port, Messaging::AsyncPort, Net::MessageClient, Net::MessageClientConnection, Messaging::Port, Net::StdTcpClientConnection, Messaging::Port, Net::StdTcpClientConnection, Messaging::Port, Net::StdTcpClient, Messaging::Port, Messaging::AsyncPort
- SendBatched() : Graphics::GraphicsInterface
- SendCreateMsg() : Graphics::GraphicsEntity
- SendMsg() : Graphics::GraphicsEntity
- SendStop() : PhysicsFeature::ActorPhysicsProperty
- SendSync() : Game::Entity
- SendTo() : Win360::Win360Socket
- SendWait() : Messaging::AsyncPort
- SequenceAction() : Actions::SequenceAction
- SerialJob() : Jobs::SerialJob
- SerialJobFuncDesc() : Jobs::SerialJobFuncDesc
- SerialJobPort() : Jobs::SerialJobPort
- SerialJobSystem() : Jobs::SerialJobSystem
- set() : Math::float4, OSX::OSXThreadLocalPtr< TYPE >, Math::float4
- Set() : Threading::SafeFlag
- set() : Math::bbox
- Set() : Debug::DebugCounter
- set() : Math::float2, Math::bbox, Math::frustum, Math::line, Math::polar
- Set() : IO::MediaType
- set() : Math::polar, Math::rectangle< TYPE >
- Set() : Util::Blob, IO::MediaType
- set() : Math::sphere
- Set() : Util::FixedTable< TYPE >
- set() : Math::sphere, Math::float4
- Set() : Util::String
- set(): Math::matrix44, Math::plane, Math::point, Math::quaternion, Math::vector
- Set(): IO::URI, IO::XmlWriter
- set_a(): Math::plane
- set_b(): Math::plane
- set_c(): Math::plane
- set_d(): Math::plane
- set_position(): Math::matrix44
- set_w(): Math::float4, Math::quaternion
- set_x(): Math::float4, Math::quaternion
- set_xaxis(): Math::matrix44
- set_y(): Math::float4, Math::quaternion
- set_yaxis(): Math::matrix44
- set_z(): Math::float4, Math::quaternion
- set_zaxis(): Math::matrix44
- SetAbsMousePos(): Input::InputEvent
- SetAccess(): Base::ResourceBase, CoreGraphics::MemoryMeshLoader, Win360::D3D9StreamMeshLoader, Base::ResourceBase, Win360::D3D9StreamMeshLoader, Base::ResourceBase
- SetAccessMode(): IO::Stream
- SetAccessPattern(): IO::Stream
- SetActionBlock(): Script::DialogTake
- SetActionList(): Actions::ActionList, Script::DialogTake
- SetActionRef(): Script::DialogTake
- SetActive(): CoreAnimation::AnimCurve
- SetActiveShaderInstance(): Base::ShaderServerBase
- SetActiveStatus(): PhysicsFeature::TriggerProperty
- SetActiveVariation(): Characters::CharacterVariationSet
- SetAdapter(): Graphics::DisplaySettings, Base::DisplayDeviceBase
- SetAddress(): Win360::Win360Socket, Net::StdTcpServer, Win360::Win360Socket
- SetAllParams(): Graphics::GlobalLightEntity
- SetAlpha(): CoreGraphics::MousePointer
- SetAlwaysOnTop(): Graphics::DisplaySettings, Base::DisplayDeviceBase
- Set AmbientLight Color(): Graphics::GlobalLightEntity, Lighting::InternalGlobalLightEntity
- SetAnimationResourceId(): Characters::CharacterNode
- SetAnimEvent(): Animation::AnimEventInfo
- SetAnimJobName(): Animation::AnimEventInfo
- SetAntiAliasQuality(): Graphics::DisplaySettings, Base::DisplayDeviceBase, Base::RenderTargetBase, Base::DisplayDeviceBase, Base::RenderTargetBase
- SetAppID(): App::Application
- SetAppName(): Interface::InterfaceHandlerBase, Core::CoreServer, Interface::InterfaceHandlerBase
- SetAppTitle(): App::Application
- SetAppVersion(): App::Application
- SetArchiveFileSystemEnabled(): IO::IoServer
- SetAspectRatio(): CoreGraphics::DisplayMode
- SetAssign(): IO::AssignRegistry
- SetAsyncEnabled(): Resources::Resource, Resources::ResourceMapper, Resources::Resource, Resources::ResourceMapper
- SetAttr(): Game::Entity
- SetAttrValue(): Game::Entity
- SetAutoManaged(): Resources::ManagedResource
- SetBackLightColor(): Graphics::GlobalLightEntity, Lighting::InternalGlobalLightEntity
- SetBackLightOffset(): Graphics::GlobalLightEntity, Lighting::InternalGlobalLightEntity
- SetBaseIndex(): CoreGraphics::PrimitiveGroup
- SetBaseTime(): Animation::AnimJob
- SetBaseVertex(): CoreGraphics::PrimitiveGroup
- SetBit(): Util::BitField< NUMBITS >
- SetBlendWeight(): Animation::AnimJob
- SetBlob(): Util::Variant, BaseGameFeature::GlobalAttrsManager, Game::Entity
- SetBlobArray(): Util::Variant
- SetBlocking(): Win360::Win360Socket, Net::StdTcpClient, Win360::Win360Socket, Net::StdTcpClient
- SetBlueprintsFilename(): BaseGameFeature::FactoryManager
- SetBool(): Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable, Util::String, Util::Variant, Direct3D9::D3D9ShaderVariable,
SetBoolArray() : Base::ShaderVariableBase, Direct3D9::D3D9ShaderVariable, Util::Variant, Direct3D9::D3D9ShaderVariable
SetBoundingBox() : Models::ModelNode, CoreGraphics::PrimitiveGroup, Models::ModelNode, Models::Model, Models::ModelNode
SetBroadcast() : Win360::Win360Socket
SetCameraEntity() : Graphics::View, InternalGraphics::InternalView, Lighting::PSSMUtil
SetCameraFocusEntity() : BaseGameFeature::FocusManager
SetCameraFocusToNextEntity() : BaseGameFeature::FocusManager
SetCameraSettings() : Graphics::CameraEntity, InternalGraphics::InternalCameraEntity
SetCanvasDimensions() : Http::SvgPageWriter
SetCapacity() : Util::RingBuffer< TYPE >
SetCastShadowsThisFrame() : Lighting::InternalAbstractLightEntity
SetCategory() : CoreAnimation::AnimEvent
SetCategoryName() : Animation::AnimEventHandlerBase
SetChar() : Input::InputEvent
SetCharPtr() : Util::String
SetClearColor() : Base::RenderTargetBase, Frame::FramePassBase, Base::MultipleRenderTargetBase, Base::RenderTargetBase, Frame::FramePassBase, Base::MultipleRenderTargetBase
SetClearDepth() : Base::RenderTargetBase, Frame::FramePassBase, Base::MultipleRenderTargetBase, Base::RenderTargetBase, Frame::FramePassBase, Base::MultipleRenderTargetBase
SetClearFlags() : Base::RenderTargetBase,
- SetDeferred() : Messaging::Message
- SetDeferredHandled() : Messaging::Message
- SetDepth() : Base::TextureBase
- SetDesc() : Http::HttpRequestHandler
- SetDescription() : Script::InfoLog, CoreGraphics::AdapterInfo
- SetDeviceId() : CoreGraphics::AdapterInfo
- SetDeviceIndex() : Input::InputEvent
- SetDeviceName() : CoreGraphics::AdapterInfo
- SetDialogLockState() : Script::DialogManager
- SetDialogTakeState() : Script::DialogManager
- SetDisplayMode() : Base::DisplayDeviceBase
- SetDisplayModeSwitchEnabled() : Base::DisplayDeviceBase, Graphics::DisplaySettings, Base::DisplayDeviceBase
- SetDriverName() : CoreGraphics::AdapterInfo
- SetDriverVersionHighPart() : CoreGraphics::AdapterInfo
- SetDriverVersionLowPart() : CoreGraphics::AdapterInfo
- SetDuration() : Animation::AnimJob
- SetElement() : Util::QuadTree< TYPE >::Node
- SetElseBlock() : Actions::IfThenElseAction
- SetEmitterAttrs() : Particles::ParticleSystemNode
- SetEmitterMeshResourceId() : Particles::ParticleSystemNode
- SetEmote() : Script::DialogTake
- SetEnabled() : PhysicsFeature::PhysicsProperty, GraphicsFeature::LightFlickerUtil, PhysicsFeature::PhysicsProperty
- SetEnqueueMode() : Animation::AnimJob
- SetEntitiesLastFrameInTrigger() : PhysicsFeature::TriggerProperty
- SetEntity() : FSM::StateMachine, Actions::IfThenElseAction, Actions::Action, Game::Property, Conditions::Condition, Game::Property, Conditions::Not, Game::Property, Actions::Action, Actions::ActionList, Actions::SequenceAction, Game::Property, Conditions::And, Conditions::Condition, Conditions::Or
- SetEntityId() : Animation::AnimEventInfo
- SetEntityMask() : Visibility::VisibilityQuery
- SetEntryDirect() : Util::SparseTable< TYPE >
- SetEnvelope() : `Particles::EmitterAttrs`
- SetEnvEntityTransform() :
  `BaseGameFeature::EnvEntityManager`
- SetExclusiveTag() : `Animation::AnimJob`
- SetFactor() : `BaseGameFeature::TimeSource`
- SetFadeInTime() : `Animation::AnimJob`
- SetFadeOutTime() : `Animation::AnimJob`
- SetFailedText() : `Script::Task`
- SetFeatureBits() : `Base::ShaderServerBase`
- SetFeatureMask() : `Base::ShaderVariationBase`
- SetFileName() : `StateObjectFeature::StateProperty`
- SetFileWriteTime() : `IO::IoServer, OSX::OSXFSWrapper, Win360::Win360FSWrapper`
- SetFirstKeyIndex() : `CoreAnimation::AnimCurve`
- SetFloat() : `Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable, Util::String, Util::Variant, Direct3D9::D3D9ShaderVariable, Base::ShaderVariableInstanceBase, Particles::EmitterAttrs, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity, IO::XmlWriter`
- SetFloat2() : `Util::String, IO::XmlWriter`
- SetFloat4() : `Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable, Util::String, Util::Variant, Direct3D9::D3D9ShaderVariable, Base::ShaderVariableInstanceBase, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity, IO::XmlWriter`
- SetFloat4Array() : `Direct3D9::D3D9ShaderVariable, Base::ShaderVariableBase, Util::Variant, Direct3D9::D3D9ShaderVariable`
- SetFloatArray() : `Base::ShaderVariableBase, Direct3D9::D3D9ShaderVariable, Util::Variant, Direct3D9::D3D9ShaderVariable`
- SetFocalLength() : `Base::TransformDeviceBase`
- SetFocusEntity() : `BaseGameFeature::FocusManager`
- `SetFocusToNextEntity()` : `BaseGameFeature::FocusManager`
- `SetFormat()` : `Base::StreamTextureSaverBase`
- `SetFragment()` : `IO::URI`
- `SetFrameId()` : `Resources::ManagedResource`
- `SetFrameShader()` : `InternalGraphics::InternalView`
- `SetFrequency()` : `GraphicsFeature::LightFlickerUtil`
- `SetFromByteOrder()` : `System::ByteOrder`
- `SetFromUInt()` : `Util::FourCC`
- `SetFullScreen()` : `Graphics::DisplaySettings`, `Base::DisplayDeviceBase`
- `SetGlobalMatrix()` : `Shared::CharJointInfo`
- `SetGraphicsEntities()` : `Visibility::VisibilityContainer`
- `SetGroup()` : `Script::DialogDesc`
- `SetGuid()` : `CoreGraphics::AdapterInfo`, `Util::Variant`, `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`, `Script::DialogDesc`, `Script::DialogTake`
- `SetGuidArray()` : `Util::Variant`
- `SetHandled()` : `Messaging::Message`
- `SetHandlerThread()` : `Messaging::AsyncPort`
- `SetHeight()` : `Base::TextureBase`, `CoreGraphics::DisplayMode`, `Base::RenderTargetBase`, `Base::TextureBase`
- `SetHighFrequencyVibrator()` : `Base::GamePadBase`
- `SetHistorySize()` : `IO::HistoryConsoleHandler`
- `SetHost()` : `IO::URI`
- `SetHostName()` : `Win360::Win360IpAddress`
- `SetHotspot()` : `CoreGraphics::MousePointer`
- `SetHour()` : `Base::CalendarTimeBase`
- `SetIconName()` : `Graphics::DisplaySettings`, `Base::DisplayDeviceBase`
- `SetId()` : `Script::DialogDesc`, `Script::DialogTake`
- `SetIndexBuffer()` : `Direct3D9::D3D9RenderDevice`, `CoreGraphics::MemoryMeshLoader`, `Base::MeshBase`, `Direct3D9::D3D9RenderDevice`, `Base::RenderDeviceBase`, `Base::MeshBase`
- `SetIndexType()` : `Base::IndexBufferBase`
- `SetInputFocusEntity()` : `BaseGameFeature::FocusManager`
- `SetInputFocusToNextEntity()` : `BaseGameFeature::FocusManager`
- SetInstanceEntity() : BaseGameFeature::CategoryManager
- SetInt() : Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, IO::XmlWriter, Util::String, Util::Variant, Direct3D9::D3D9ShaderVariable, Base::ShaderVariableInstanceBase, Particles::EmitterAttrs, Direct3D9::D3D9ShaderVariable, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- SetIntArray() : Base::ShaderVariableBase, Direct3D9::D3D9ShaderVariable, Util::Variant, Direct3D9::D3D9ShaderVariable
- SetIntensityAmplitude() : GraphicsFeature::LightFlickerUtil
- SetInViewSpace() : Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance
- SetJointName() : Shared::CharJointInfo
- SetKeepAlive() : Win360::Win360Socket
- SetKey() : Input::InputEvent
- SetKeyDuration() : CoreAnimation::AnimClip
- SetKeyStride() : CoreAnimation::AnimClip
- SetLevelName() : BaseGameFeature::GameStateHandler
- SetLightDir() : Lighting::PSSMUtil
- SetLightEntity() : GraphicsFeature::LightFlickerUtil
- SetLightingMode() : Frame::FrameBatch
- SetLightType() : Lighting::InternalAbstractLightEntity, Graphics::AbstractLightEntity, Lighting::InternalAbstractLightEntity, Graphics::AbstractLightEntity
- SetLoader() : Resources::Resource
- SetLocalBoundingBox() : InternalGraphics::InternalGraphicsEntity
- SetLocalizeDialog() : Script::DialogManager
- SetLocalMatrix() : Shared::CharJointInfo
- SetLocalPath() : IO::URI
- SetLocked() : Script::DialogDesc
- SetLockedToViewer() : Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode,
Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance

- SetLogFileEnabled() : App::RenderApplication
- SetLogLevel() : Script::InfoLog
- SetLoopCount() : Animation::PlayClipJob
- SetLowFrequencyVibrator() : Base::GamePadBase
- SetMainRenderTarget() : Frame::FrameShader
- SetManagedResource() : Resources::LoadingResource
- SetManagedResourceClass() : Resources::SimpleResourceMapper
- SetMapper() : Resources::ResourceScheduler, Resources::TexturePoolMapperScheduler
- SetMatrix() : Direct3D9::D3D9ShaderVariable, Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable, Base::ShaderVariableInstanceBase
- SetMatrix44() : Util::String, Util::Variant, BaseGameFeature::GlobalAttrsManager, Game::Entity, IO::XmlWriter
- SetMatrix44Array() : Util::Variant
- SetMatrixArray() : Base::ShaderVariableBase, Direct3D9::D3D9ShaderVariable
- SetMaxDistance() : Models::TransformNode
- SetMaxNumLocalPlayers() : Base::InputServerBase
- SetMaxProgressValue() : BaseGameFeature::LoaderServer
- SetMaxTriggerDistance() : BaseGameFeature::EntityManager
- SetMediaType() : IO::Stream
- SetMemoryMappingEnabled() : IO::BinaryWriter, IO::BinaryReader
- SetMemoryOffset() : Base::RenderTargetBase
- SetMeshResourceId() : Models::ShapeNode
- SetMeshResourceLoader() : Models::ShapeNode
- SetMethod() : Http::HttpRequest, Http::HttpRequestWriter
- SetMilliSecond() : Base::CalendarTimeBase
- SetMinDistance() : Models::TransformNode
- SetMinute() : Base::CalendarTimeBase
- SetMipLevel() : Base::StreamTextureSaverBase
- SetMipMapsEnabled(): `Base::RenderTargetBase`
- SetModelEntity(): `Models::ModellInstance`
- SetModelResourceMapper(): `Models::ModelServer`
- SetModelTransform(): `Base::TransformDeviceBase`
- SetMonth(): `Base::CalendarTimeBase`
- SetMountStandardArchivesEnabled(): `App::RenderApplication`
- SetMouseButton(): `Input::InputEvent`
- SetMouseMovement(): `RenderUtil::MayaCameraUtil`
- SetMRTIndex(): `Base::RenderTargetBase`
- SetMultipleRenderTarget(): `Frame::FramePassBase`, `InternalGraphics::InternalView`, `Frame::FramePassBase`
- SetMyThreadName(): `Win360::Win360Thread`
- SetName(): `Win360::Win360Thread`, `FSM::StateMachine`, `Http::HttpRequestHandler`, `OSX::OSXThread`, `Base::ShaderVariationBase`, `Animation::AnimJob`, `Base::ShaderVariableBase`, `Win360::Win360Thread`, `Models::ModelNode`, `Win360::Win360Thread`, `Models::ModelNode`, `FSM::State`, `Models::ModelNode`, `Frame::FramePassBase`, `Frame::FrameShader`, `Win360::Win360Thread`, `InternalGraphics::InternalStage`, `InternalGraphics::InternalView`, `Models::ModelNode`, `Http::HttpRequestHandler`, `Base::ShaderVariableBase`, `Base::ShaderVariationBase`, `Animation::AnimJob`, `Base::ShaderVariationBase`, `Models::ModelNode`, `Http::HttpRequestHandler`, `Characters::CharacterSkinList`, `Http::HttpRequestHandler`, `Frame::FramePassBase`, `Win360::Win360Thread`, `BaseGameFeature::UserProfile`, `Win360::Win360Thread`, `CoreAnimation::AnimClip`, `Win360::Win360Thread`, `Http::HttpRequestHandler`, `CoreAnimation::AnimEvent`
- SetNodeFilter(): `Frame::FrameBatch`
- SetNoDelay(): `Win360::Win360Socket`
- SetNormMousePos(): `Input::InputEvent`
- SetNumCurves(): `CoreAnimation::AnimClip`
- SetNumIndices(): `CoreGraphics::PrimitiveGroup`, `Base::IndexBufferBase`
- SetNumKeys(): `CoreAnimation::AnimClip`
- SetNumMipLevels(): `Base::TextureBase`
- SetNumPasses(): `Base::ShaderVariationBase`
- `SetNumVertices`: `Base::VertexBufferBase`, `CoreGraphics::PrimitiveGroup`, `Base::VertexBufferBase`
- `SetObject`: `Util::Variant`
- `SetObserver`: `Visibility::VisibilityQuery`
- `setoffset`: `Math::transform44`
- `SetOffsetMatrix`: `Models::TransformNodeInstance`
- `SetOpenText`: `Script::Task`
- `SetOrbitButton`: `RenderUtil::MayaCameraUtil`
- `SetOrbiting`: `RenderUtil::MayaCameraUtil`
- `SetOrientation`: `CoreGraphics::MousePointer`
- `SetOverrideDefaultRenderTarget`: `Base::RenderDeviceBase`
- `SetOverrideRootDirectory`: `App::RenderApplication`
- `SetPanButton`: `RenderUtil::MayaCameraUtil`
- `SetPanning`: `RenderUtil::MayaCameraUtil`
- `SetParent`: `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`
- `SetParentKey`: `Script::Task`
- `SetParentWindow`: `Base::DisplayDeviceBase`, `Graphics::Display`, `Base::DisplayDeviceBase`
- `SetPerfHUDEnabled`: `Debug::DebugGraphicsHandler`
- `SetPixelFormat`: `Base::TextureBase`, `CoreGraphics::DisplayMode`, `Base::TextureBase`
- `SetPlaceholder`: `Resources::ManagedResource`
- `SetPlaceholderResourceId`: `Resources::ResourceMapper`
- `SetPlayerIndex`: `Base::GamePadBase`
- `SetPort`: `Http::HttpServer`, `IO::URI`, `Win360::Win360IpAddress`
- `SetPosition`: `Models::TransformNodeInstance`, `Models::TransformNode`, `Models::TransformNodeInstance`, `Models::TransformNode`, `CoreGraphics::MousePointer`, `Models::TransformNode`, `Models::TransformNodeInstance`
- `setPosition`: `Math::transform44`
- `SetPosition`: `Models::TransformNodeInstance`, `Models::TransformNode`, `Models::TransformNodeInstance`, `Models::TransformNode`
- `setPositionAmplitude`: `GraphicsFeature::LightFlickerUtil`
- SetPostInfinityType() : CoreAnimation::AnimClip
- SetPreInfinityType() : CoreAnimation::AnimClip
- SetPrimitiveGroup() : Base::RenderDeviceBase
- SetPrimitiveGroupIndex() : Models::ShapeNode, Particles::ParticleSystemNode
- SetPrimitiveGroups() : CoreGraphics::MemoryMeshLoader, Base::MeshBase
- SetPrimitiveTopology() : CoreGraphics::PrimitiveGroup
- SetPriority() : Win360::Win360Thread, OSX::OSXThread, Win360::Win360Thread, Resources::ManagedResource, Resources::LoadingResource, Resources::ManagedResource, Win360::Win360Thread, Resources::ManagedResource, Resources::LoadingResource, Win360::Win360Thread
- SetProgressResource() : BaseGameFeature::LoaderServer
- SetProgressText() : BaseGameFeature::LoaderServer
- SetProjMapUvOffsetAndScale() : Lighting::InternalAbstractLightEntity, Graphics::AbstractLightEntity, Lighting::InternalAbstractLightEntity, Graphics::AbstractLightEntity
- SetProjTransform() : Base::TransformDeviceBase
- SetQuadTreeSettings() : Visibility::VisibilityQuadtree
- SetQuery() : IO::URI
- SetQuitRequested() : App::RenderApplication, Game::GameServer, Base::InputServerBase, App::RenderApplication, Base::InputServerBase
- SetReadOnly() : OSX::OSXFSWrapper, IO::IoServer, Win360::Win360FSWrapper
- SetRecvBufSize() : Win360::Win360Socket
- SetRenderDebug() : Game::FeatureUnit, InternalGraphics::InternalGraphicsServer, Game::FeatureUnit
- SetRenderTarget() : Frame::FramePassBase, InternalGraphics::InternalView, Frame::FramePassBase
- SetResolved() : Models::VisResolveContainer< TYPE >
- SetResolveDepthTextureResourceId() : Base::RenderTargetBase
- SetResolveRect() : Base::RenderTargetBase
SetResolveTargetCpuAccess() : Base::RenderTargetBase
SetResolveTextureHeight() : Base::RenderTargetBase
SetResolveTextureResourceId() : Base::RenderTargetBase
SetResolveTextureWidth() : Base::RenderTargetBase
SetResource() : Resources::ManagedResource
SetResourceClass() : Resources::SimpleResourceMapper
SetResourceCreatorClass() : Resources::PoolResourceMapper
SetResourceId() : Resources::ManagedResource,
CoreGraphics::MousePointer, Resources::Resource,
Resources::ManagedResource, Graphics::ModelEntity,
Resources::Resource, InternalGraphics::InternalModelEntity,
Resources::Resource, Resources::ManagedResource,
Resources::Resource, Resources::ManagedResource,
Resources::Resource
SetResourceLoaderClass() :
Resources::SimpleResourceMapper
SetResourceMappers() : Graphics::Display
SetResourceStreamingLevelOfDetail() : Models::ModelNode
SetResourceType() : Resources::ManagedResource
SetResponseContentStream() : Http::HttpRequest
SetResult() : Conditions::Condition
SetReturnType() : App::Application
SetReUseAddr() : Win360::Win360Socket
SetReuseTexture() : Resources::D3D9TextureStreamer
SetRevision() : CoreGraphics::AdapterInfo
SetRootDirectory() : Core::CoreServer
SetRootLocation() : Http::HttpRequestHandler
SetRootNodeOffsetMatrix() : Graphics::ModelEntity,
InternalGraphics::InternalModelEntity
SetRootNodePath() : Graphics::ModelEntity,
InternalGraphics::InternalModelEntity
SetRotate() : Models::TransformNodeInstance
setrotate() : Math::transform44
SetRotate() : Models::TransformNodeInstance
SetRotatePivot() : Models::TransformNode,
Models::TransformNodeInstance, Models::TransformNode,
Models::TransformNodeInstance, Models::TransformNode,
Models::TransformNodeInstance
setrotatetepivot() : Math::transform44
- SetRotatePivot() : Models::TransformNode, Models::TransformNodeInstance
- SetRotation() : Models::TransformNode
- setrow0() : Math::matrix44
- setrow1() : Math::matrix44
- setrow2() : Math::matrix44
- setrow3() : Math::matrix44
- SetSaveActionsFlag() : Actions::SequenceAction
- SetSaveGame() : BaseGameFeature::GameStateHandler
- SetSaver() : Resources::Resource
- SetScale() : Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance
- setscale() : Math::transform44
- SetScale() : Models::TransformNode, Models::TransformNodeInstance
- SetScalePivot() : Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance
- setscalepivot() : Math::transform44
- SetScalePivot() : Models::TransformNode, Models::TransformNodeInstance
- SetScheme() : IO::URI
- SetSecond() : Base::CalendarTimeBase
- SetSemantic() : Base::ShaderVariableBase
- SetSendBufSize() : Win360::Win360Socket
- SetServerAddress() : Net::StdTcpClient
- SetSetupMode() : BaseGameFeature::GameStateHandler
- SetShader() : Models::StateNode, Frame::FramePassBase, Models::StateNode, Frame::FrameBatch, Frame::FramePassBase
- SetShaderFeatures() : Frame::FrameBatch
- SetShadowBufferUvOffsetAndScale() : Lighting::InternalAbstractLightEntity
- SetShadowIntensity() : Lighting::InternalAbstractLightEntity
- SetShadowTransform() : Lighting::InternalAbstractLightEntity
- SetSharedData() : InternalGraphics::InternalGraphicsEntity
- SetShortText() : Script::DialogTake
- SetSignalOnEnqueueEnabled() : Threading::SafeQueue< TYPE
SetSingleThreadMode() : Http::HttpServer
SetSize() : IO::Stream,
    Resources::ResourceDictionary::Entry,
    CoreGraphics::MousePointer, IO::MemoryStream,
    Util::FixedArray< TYPE >, Util::FixedTable< TYPE >,
    IO::Stream
SetSkins() : Characters::CharacterSkinList
SetSkippedMips() : Base::TextureBase
SetSortingMode() : Frame::FrameBatch
SetSound() : Script::DialogTake
SetSource() : Script::InfoLog
SetSourceSlot() : Resources::PoolLoadingResource
SetSpeaker() : Script::DialogDesc, Script::DialogTake
SetStackSize() : Win360::Win360Thread, OSX::OSXThread,
    Win360::Win360Thread
SetStage() : InternalGraphics::InternalView
SetStartKeyIndex() : CoreAnimation::AnimClip
SetStartTime() : Animation::AnimJob
SetState() : Resources::ResourceLoader,
    Resources::Resource, Resources::ResourceLoader,
    Resources::Resource, Resources::ResourceLoader,
    Resources::Resource, Resources::ResourceLoader,
    Resources::Resource, Resources::ResourceLoader,
    Resources::Resource, Resources::ResourceLoader,
    Resources::Resource, Resources::ResourceLoader,
    Resources::Resource, App::GameApplication,
    Resources::Resource
SetStatic() : CoreAnimation::AnimCurve
SetStaticKey() : CoreAnimation::AnimCurve
SetStatus() : Http::HttpRequest
SetStatusCode() : Http::HttpResponseWriter
SetStream() : IO::StreamReader,
    Base::StreamTextureSaverBase, IO::StreamWriter,
    Base::StreamTextureSaverBase, IO::StreamReader,
    Messaging::MessageReader, IO::StreamReader,
    IO::StreamWriter, IO::StreamReader,
    Base::StreamTextureSaverBase, IO::StreamWriter,
    Messaging::MessageWriter, IO::StreamWriter,
IO::StreamReader, IO::StreamWriter
- SetStreamByteOrder(): IO::BinaryReader, IO::BinaryWriter
- SetStreamMeshLoader(): Models::StreamModelLoader
- SetStreamSource(): Direct3D9::D3D9RenderDevice, Base::RenderDeviceBase
- SetString(): BaseGameFeature::GlobalAttrsManager, Http::SvgPageWriter, BaseGameFeature::UserProfile, Script::ActionReader, Game::Entity, Util::Variant, Http::SvgPageWriter, IO::XmlWriter
- SetStringArray(): Util::Variant
- SetStringAttr(): Models::ModelNode
- SetSubSystemId(): CoreGraphics::AdapterInfo
- SetTargetResource(): Resources::LoadingResource
- SetTargetSlot(): Resources::PoolLoadingResource
- SetTargetState(): FSM::Transition
- SetTaskStatusUnchanged(): Script::Task
- SetTaskViewed(): Script::Task
- SetTexture(): Direct3D9::D3D9ShaderVariable, Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable, Base::ShaderVariableInstanceBase
- SetThenBlock(): Actions::IfThenElseAction
- SetTime(): CoreAnimation::AnimEvent
- SetTimeFactor(): Animation::AnimJob, BaseGameFeature::TimeManager, FrameSync::FrameSyncTimer
- SetTimeOffset(): Animation::AnimJob
- SetTimeStamp(): Script::DialogTake
- Settings(): Graphics::Display
- setTitle(): Base::GameContentServerBase, Http::HtmlPageWriter, Base::GameContentServerBase
- setTitleId(): Base::GameContentServerBase
- SetToByteOrder(): System::ByteOrder
- SetToDefault(): BaseGameFeature::UserProfile
- SetToFirstChild(): IO::BXmlReader, IO::BXmlLoaderUtil, IO::XmlReader
- SetToNextChild(): IO::BXmlLoaderUtil, IO::BXmlReader, IO::XmlReader
- SetToNode() : `IO::BXmlReader`, `IO::XmlReader`
- SetToParent() : `IO::BXmlReader`, `IO::XmlReader`, `IO::BXmlLoaderUtil`
- SetTrackIndex() : `Animation::AnimJob`
- SetTransformFromPosDirAndRange() : `Graphics::PointLightEntity`
- SetTransformFromPosDirRangeAndCone() : `Graphics::SpotLightEntity`
- SetTripleBufferingEnabled() : `Graphics::DisplaySettings`, `Base::DisplayDeviceBase`
- SetUnitDimensions() : `Http::SvgPageWriter`
OSX::SysFunc, RenderUtil::MayaCameraUtil, Jobs::SerialJobSystem, Base::GameContentServerBase, Base::MemoryVertexBufferLoaderBase, Base::JobPortBase, Jobs::TPJobSlice, Base::MemoryVertexBufferLoaderBase, Base::JobPortBase, Characters::CharacterAnimationController, Base::MemoryIndexBufferLoaderBase, IO::AssignRegistry, Direct3D9::D3D9RenderTarget, Win360::Win360MemoryPool, Debug::DebugCounter, Jobs::TPJobSystem, Debug::DebugTimer, Particles::EnvelopeCurve, Base::MemoryIndexBufferLoaderBase, RenderModules::RTPluginRegistry, Util::QuadTree< TYPE >::Node, Frame::FramePostEffect, Memory::PoolArrayAllocator, Visibility::ObserverContext, Base::JobPortBase, Win360::D3D9VertexLayout, IO::SchemeRegistry, Direct3D9::D3D9ParticleRenderer, Base::JobPortBase, OSX::OSXMemoryPool, IO::ZipFileSystem, IO::ZipArchive, Models::ModelNodeInstance, Base::VertexLayoutBase, Base::JobBase, Base::GameContentServerBase, Particles::ParticleSystem, Base::MemoryVertexBufferLoaderBase, IO::ZipArchive, Win360::Win360ThreadBarrier, Base::SkinnedMeshRendererBase, Graphics::CameraEntity, Base::ShaderVariableInstanceBase, Win360::D3D9VertexLayout, Base::ParticleRendererBase, Models::TransformNodeInstance, Direct3D9::D3D9ParticleRenderer, RenderModules::RenderModule, RenderUtil::DrawFullScreenQuad, Characters::Character, Base::ShaderInstanceBase, Base::ShaderVariableInstanceBase, Particles::EnvelopeSampleBuffer, Characters::CharacterSkeleton, Characters::CharacterServer, Characters::CharacterNodeInstance, OSX::OSXHeap, Base::MouseRenderDeviceBase, Models::ModellInstance, Util::Blob, Base::JobBase, Direct3D9::D3D9ShaderInstance, Characters::CharacterInstance,
Models::TransformNodeInstance, IO::ZipFileSystem, Graphics::GraphicsEntity, Base::MemoryIndexBufferLoaderBase, Models::TransformNodeInstance, Win360::Win360Heap, Direct3D9::D3D9RenderTarget, IO::ArchiveBase, Base::MouseRenderDeviceBase, Characters::CharacterVariationSet, Characters::CharacterJoint, Win32::SysFunc, Characters::CharacterSkinNodeInstance, Characters::CharacterSkinSet, Util::StringBuffer, Particles::ParticleSystemNodeInstance, Base::SkinnedMeshRendererBase, Characters::CharacterSkeletonInstance, CoreAnimation::AnimSampleBuffer, CoreAnimation::AnimKeyBuffer, FrameSync::FrameSyncTimer, Animation::AnimSequencer

- setup_from_point_and_normal() : Math::plane
- setup_from_points() : Math::plane
- SetupAcceptedMessages() : PhysicsFeature::PhysicsProperty, PhysicsFeature::ActorPhysicsProperty, Messaging::Port, GraphicsFeature::AnimationControlProperty, Messaging::Port, GraphicsFeature::CameraProperty, GraphicsFeature::ActorGraphicsProperty, Messaging::Port, GraphicsFeature::MayaCameraProperty, Messaging::Port, Script::DialogManager, Messaging::Port, GraphicsFeature::ChaseCameraProperty, Messaging::Port, StateObjectFeature::StateGraphicsProperty, GraphicsFeature::GraphicsProperty, BaseGameFeature::TransformableProperty, StateObjectFeature::StateProperty, Messaging::Port, PhysicsFeature::TriggerProperty, Messaging::Port

- SetupAnimDrivenMotion() : Characters::CharacterAnimationController
- SetupAppFromCmdLineArgs() : App::GameApplication
- SetupAttributes() : BaseGameFeature::FactoryManager
- SetupCallbacks() : PhysicsFeature::PhysicsProperty, PhysicsFeature::MouseGripperProperty, Game::Property, GraphicsFeature::GraphicsProperty, Game::Property, PhysicsFeature::TriggerProperty,
GraphicsFeature::GraphicsProperty,
GraphicsFeature::CameraProperty,
GraphicsFeature::GraphicsProperty,
GraphicsFeature::MayaCameraProperty,
StateObjectFeature::StateProperty, Game::Property,
GraphicsFeature::GraphicsProperty,
GraphicsFeature::CameraProperty,
PhysicsFeature::ActorPhysicsProperty

- SetupDefaultAttributes() : Game::Property,
  GraphicsFeature::MayaCameraProperty,
  PhysicsFeature::TriggerProperty,
  GraphicsFeature::ChaseCameraProperty,
  GraphicsFeature::CameraProperty,
  StateObjectFeature::StateGraphicsProperty,
  GraphicsFeature::AnimationControlProperty,
  Game::Property, GraphicsFeature::ActorGraphicsProperty,
  Game::Property, PhysicsFeature::ActorPhysicsProperty,
  GraphicsFeature::CameraProperty,
  GraphicsFeature::GraphicsProperty,
  PhysicsFeature::MouseGripperProperty,
  StateObjectFeature::StateProperty,
  BaseGameFeature::TransformableProperty,
  PhysicsFeature::PhysicsProperty

- SetupFromD3D9CubeTexture() : Direct3D9::D3D9Texture
- SetupFromD3D9Texture() : Direct3D9::D3D9Texture
- SetupFromD3D9VolumeTexture() : Direct3D9::D3D9Texture
- SetupFromFileInMemory() : IO::BXmlLoaderUtil
- SetupFromNax3() : CoreAnimation::StreamAnimationLoader
- SetupGameFeatures() : App::GameApplication
- SetupGraphicsEntities() : GraphicsFeature::GraphicsProperty,
  GraphicsFeature::ActorGraphicsProperty
- SetupIndexedPrimitives() : CoreGraphics::RenderShape
- SetupJoint() : Characters::CharacterSkeleton
- SetupManagedTextureVariable() : Models::StateNode
- SetupMeshFromMemory() :
  CoreGraphics::MemoryMeshLoader
- SetupMultiSampleType() : Direct3D9::D3D9RenderTarget
- SetupOrthogonal() : Shared::CameraSettings
- SetupPerspectiveFov() : Shared::CameraSettings
- SetupPools() : Resources::PoolResourceMapper
- SetupPrimitives() : CoreGraphics::RenderShape
- SetupResourceFromStream() :
  CoreAnimation::StreamAnimationLoader,
  Resources::D3D9TextureStreamer,
  Resources::StreamResourceLoader
- SetupRun() : Jobs::TPJobCommand
- SetupSimpleShape() : CoreGraphics::RenderShape
- SetupStateHandlers() : App::GameApplication
- SetupSync() : Jobs::TPJobCommand
- SetupTexture2DFromStream() :
  Resources::D3D9TextureStreamer
- SetupTextureCubeFromStream() :
  Resources::D3D9TextureStreamer
- SetupXAxis() : Http::SvgLineChartWriter
- SetupYAxis() : Http::SvgLineChartWriter
- SetURI() : IO::Stream, Http::HttpRequestWriter, Http::HttpRequest, Resources::ResourceDictionary, IO::Stream
- SetUsage() : Win360::D3D9StreamMeshLoader,
  Base::ResourceBase, CoreGraphics::MemoryMeshLoader,
  Base::ResourceBase, Win360::D3D9StreamMeshLoader,
  Base::ResourceBase
- SetUserAgent() : Http::HttpRequestWriter
- SetUserInfo() : IO::URI
- SetUserProfile() : BaseGameFeature::LoaderServer
- SetValid() : InternalGraphics::InternalGraphicsEntity
- SetValue() : Base::ShaderVariableInstanceBase
- SetVariationSetName() : Characters::CharacterInstance
- SetVendorId() : CoreGraphics::AdapterInfo
- SetVersion() : Base::GameContentServerBase
- SetVertexBuffer() : Base::MeshBase,
  CoreGraphics::MemoryMeshLoader
- SetVertexLayout() : Base::RenderDeviceBase,
  Base::VertexBufferBase, Direct3D9::D3D9RenderDevice,
  Base::VertexBufferBase, Direct3D9::D3D9RenderDevice
- SetVerticalSyncEnabled() : Base::DisplayDeviceBase,
  Graphics::DisplaySettings, Base::DisplayDeviceBase
- SetViewTransform() : Base::TransformDeviceBase
- SetVisibilityContexts() : Visibility::VisibilityContainer
- SetVisibilityOnAttachments() : InternalGraphics::AttachmentServer
- SetVisibleFrameId() : Visibility::VisibilityContext
- SetVisualizeMipMaps() : Base::RenderDeviceBase
- SetWaitTimeout() : Messaging::BlockingHandlerThread
- SetWeekday() : Base::CalendarTimeBase
- SetWeight() : Animation::AnimEventInfo
- SetWidth() : Base::TextureBase, Base::RenderTargetBase, Base::TextureBase, Base::RenderTargetBase, Base::TextureBase, Base::RenderTargetBase, CoreGraphics::DisplayMode
- SetWindowTitle() : Base::DisplayDeviceBase, Graphics::DisplaySettings
- SetXPos() : CoreGraphics::DisplayMode
- SetYear() : Base::CalendarTimeBase
- SetYPos() : CoreGraphics::DisplayMode
- SetZoomButton() : RenderUtil::MayaCameraUtil
- SetZoomIn() : RenderUtil::MayaCameraUtil
- SetZoomInButton() : RenderUtil::MayaCameraUtil
- SetZoomOut() : RenderUtil::MayaCameraUtil
- SetZoomOutButton() : RenderUtil::MayaCameraUtil
ShaderBase() : Base::ShaderBase
ShaderInstanceBase() : Base::ShaderInstanceBase
ShaderPageHandler() : Debug::ShaderPageHandler
ShaderServer() : CoreGraphics::ShaderServer
ShaderServerBase() : Base::ShaderServerBase
ShaderVariableBase() : Base::ShaderVariableBase
ShaderVariableInstanceBase() :
  Base::ShaderVariableInstanceBase
ShaderVariationBase() : Base::ShaderVariationBase
ShadowServer() : Lighting::ShadowServer
ShapeNode() : Models::ShapeNode
ShapeNodeInstance() : Models::ShapeNodeInstance
ShapeRenderer() : CoreGraphics::ShapeRenderer
ShapeRendererBase() : Base::ShapeRendererBase
ShowActionInfo() : Actions::Action
Shutdown() : Net::StdTcpClientConnection, Net::MessageClientConnection, Util::Blob
Signal() : Threading::SafePriorityQueue< PRITYPE, TYPE >, Win360::Win360Event, Threading::SafeQueue< TYPE >
SignalContinue() : Win360::Win360ThreadBarrier
SignalPendingResourceEvent() : Graphics::GraphicsInterface
SimpleResourceMapper() :
  Resources::SimpleResourceMapper
SimpleTree() : Util::SimpleTree< VALUETYPE >
Size() : Util::Array< TYPE >, Util::FixedArray< TYPE >, Util::Queue< TYPE >, Util::Blob,
  Threading::SafePriorityQueue< PRITYPE, TYPE >, Util::List< TYPE >, Util::RingBuffer< TYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >
size() : Math::bbox
Size() : Util::Stack< TYPE >, Util::List< TYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::Array< TYPE >,
  Threading::SafeQueue< TYPE >
SizeOf() : CoreGraphics::IndexType
Skeleton() : Characters::CharacterInstance, Characters::Character
SkinLibrary() : Characters::Character
SkinnedMeshRenderer() : Characters::SkinnedMeshRenderer
- SkinnedMeshRendererBase() : Base::SkinnedMeshRendererBase
- SkinSet() : Characters::CharacterInstance
- SkipSegments() : PhysicsFeature::ActorPhysicsProperty
- Sleep() : Win32::SysFunc, OSX::SysFunc
- slerp() : Math::quaternion
- SM30LightServer() : Lighting::SM30LightServer
- SM30ShadowServer() : Lighting::SM30ShadowServer
- Sort() : Util::FixedArray< TYPE >, Util::Array< TYPE >
- SortIfDirty() : Util::Dictionary< KEYTYPE, VALUETYPE >
- soundName : StateObjectFeature::StateInfo
- SparseTable() : Util::SparseTable< TYPE >
- sphere() : Math::sphere
- splat() : Math::float4
- splat_w() : Math::float4
- splat_x() : Math::float4
- splat_y() : Math::float4
- splat_z() : Math::float4
- SpotLightEntity() : Graphics::SpotLightEntity
- squad() : Math::quaternion
- squadsetup() : Math::quaternion
- Stack() : Util::Stack< TYPE >
- Start() : Win360::Win360Thread, Game::GameServer, Actions::Action, Win360::Win360Thread, Debug::DebugTimer, Win360::Win360Thread, Actions::Action, Win360::Win360Timer, OSX::OSXThread, Win360::Win360Thread
- start() : Math::line
- Start() : Actions::Action, Win360::Win360Thread, Actions::Action, Actions::SequenceAction, Win360::Win360Thread
- StartAccum() : Debug::DebugTimer
- StartAsyncEvaluation() : Animation::AnimSequencer
- StartCurrAction() : Actions::SequenceAction
- StartDialog() : Script::DialogManager
- StartEntities() : BaseGameFeature::EntityManager
- StartFixedFrameTime() : FrameSync::FrameSyncHandlerThread
- StartRenderDebug() : Game::FeatureUnit
- **StartTime()** : `FrameSync::FrameSyncTimer`
- **StartTimeEffect()** : `BaseGameFeature::TimeManager`
- **StartUpdateCharacterSkeletons()** :
  - `Characters::CharacterServer`
- **State** : `Resources::Resource`, `FSM::State`
- **StateGraphicsProperty()** :
  - `StateObjectFeature::StateGraphicsProperty`
- **StateMachine()** :
  - `FSM::StateMachine`
- **StateNode()** :
  - `Models::StateNode`
- **StateNodeInstance()** :
  - `Models::StateNodeInstance`
- **StateProperty()** :
  - `StateObjectFeature::StateProperty`
- **StateToString()** :
  - `Script::Task`
- **StdTcpClient()** :
  - `Net::StdTcpClient`
- **StdTcpClientConnection()** :
  - `Net::StdTcpClientConnection`
- **StdTcpServer()** :
  - `Net::StdTcpServer`
- **StepTime** :
  - `Particles::ParticleSystem`
- **Stop()** :
  - `Win360::Win360Timer`, `Game::GameServer`,
  - `Actions::Action`, `Animation::AnimJob`, `Debug::DebugTimer`,
  - `Actions::Action`, `OSX::OSXThread`, `Win360::Win360Thread`,
  - `Actions::Action`, `Jobs::TPWorkerThread`,
  - `Win360::Win360Thread`,
  - `PhysicsFeature::ActorPhysicsProperty`,
  - `Win360::Win360Thread`, `Actions::Action`,
  - `Win360::Win360Thread`, `Actions::SequenceAction`
- **StopAccum()** :
  - `Debug::DebugTimer`
- **StopAllTracks()** :
  - `Characters::CharacterAnimationController`,
  - `Animation::AnimSequencer`
- **StopFixedFrameTime()** :
  - `FrameSync::FrameSyncHandlerThread`
- **StopRenderDebug()** :
  - `Game::FeatureUnit`
- **StopTime()** :
  - `FrameSync::FrameSyncTimer`
- **StopTimeEffect()** :
  - `BaseGameFeature::TimeManager`
- **StopTrack()** :
  - `Animation::AnimSequencer`,
  - `Characters::CharacterAnimationController`
- **store()** :
  - `Math::quaternion`, `Math::float4`, `Math::matrix44`,
  - `Math::float4`
- **storeu()** :
  - `Math::float4`, `Math::matrix44`, `Math::quaternion`,
  - `Math::float4`
- **StrChr()** :
  - `Util::String`
- StrCmp() : Util::String
- Stream() : IO::Stream
- stream() : Math::quaternion, Math::float4, Math::matrix44, Math::float4
- StreamingTexturePageHandler() : Debug::StreamingTexturePageHandler
- StreamModelLoader() : Models::StreamModelLoader
- StreamReader() : IO::StreamReader
- StreamResourceLoader() : Resources::StreamResourceLoader
- StreamTextureSaverBase() : Base::StreamTextureSaverBase
- StreamWriter() : IO::StreamWriter
- String() : Util::String
- StringAtom() : Util::StringAtom
- StringAtomTableBase() : Util::StringAtomTableBase
- StringBuffer() : Util::StringBuffer
- StringToLogLevel() : Script::InfoLog
- StringToMonth() : Base::CalendarTimeBase
- StringToState() : Script::Task
- StringToType() : Base::ShaderVariableBase, Script::DialogTake, Util::Variant
- StringToWeekday() : Base::CalendarTimeBase
- Strip() : Util::String
- StripFileExtension() : Util::String
- StrLen() : Util::String
- SubstituteChar() : Util::String
- SubstituteString() : Util::String
- SubstitutionManager() : Script::SubstitutionManager
- SupportsDisplayMode() : Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice, Base::DisplayDeviceBase, Graphics::Display, Direct3D9::D3D9DisplayDevice
- SvgLineChartWriter() : Http::SvgLineChartWriter
- SvgPageWriter() : Http::SvgPageWriter
- SvgTestPageHandler() : Debug::SvgTestPageHandler
- SwitchEntity() : InternalGraphics::AttachmentServer
- SwitchFocusEntities() : BaseGameFeature::FocusManager
- SystemInfoBase() : Base::SystemInfoBase
- SystemTimeSource() : BaseGameFeature::SystemTimeSource
- SystemTimeToFileTime() : Win360::Win360CalendarTime, Base::CalendarTimeBase
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- t -

- TableRow2() : [Http::HtmlPageWriter](#)
- TalkTo() : [Script::DialogManager](#)
- targetResource : [Resources::LoadingResource](#)
- Task() : [Script::Task](#)
- TcpMessageCodec() : [Net::TcpMessageCodec](#)
- Tell() : [OSX::OSXFSWrapper](#), [Win360::Win360FSWrapper](#)
- TerminateAtIndex() : [Util::String](#)
- Test() : [Threading::SafeFlag](#)
- TestAndClearIfSet() : [Threading::SafeFlag](#)
- Text() : [Http::SvgPageWriter](#), [Http::HtmlPageWriter](#)
- TextElement() : [CoreGraphics::TextElement](#)
- TextReader() : [IO::TextWriter](#)
- TextRenderer() : [CoreGraphics::TextRenderer](#)
- TextRendererBase() : [Base::TextRendererBase](#)
- TextureBase() : [Base::TextureBase](#)
- TexturePageHandler() : [Debug::TexturePageHandler](#)
- TexturePoolMapperScheduler() : [Resources::TexturePoolMapperScheduler](#)
- TextWriter() : [IO::TextWriter](#)
- ThreadCloseHandlers() : [Messaging::HandlerThreadBase](#)
- ThreadDiscardDeferredMessages() : [Messaging::HandlerThreadBase](#)
- ThreadHandleMessages() : [Messaging::HandlerThreadBase](#)
- ThreadHandleSingleMessage() : Messaging::HandlerThreadBase
- ThreadOpenHandlers() : Messaging::HandlerThreadBase
- ThreadPageHandler() : Debug::ThreadPageHandler
- ThreadSafeDisplayEventHandler() : CoreGraphics::ThreadSafeDisplayEventHandler
- ThreadSafeRenderEventHandler() : CoreGraphics::ThreadSafeRenderEventHandler
- ThreadSignalMessageHandled() : Messaging::HandlerThreadBase
- ThreadStopRequested() : Win360::Win360Thread, OSX::OSXThread, Win360::Win360Thread
- ThreadUpdateDeferredMessages() : Messaging::HandlerThreadBase
- ThreadUpdateHandlers() : Messaging::HandlerThreadBase
- TimeManager() : BaseGameFeature::TimeManager
- TimeSource() : BaseGameFeature::TimeSource
- to_axisangle() : Math::quaternion
- to_matrix44() : Math::bbox
- TogglePerfHUD() : Debug::DebugGraphicsHandler
- ToggleSkin() : Characters::CharacterSkinSet
- ToHtml() : Http::HtmlElement
- ToHumanReadableString() : Http::HttpStatus
- Tokenize() : Util::String
- ToLower() : Util::String
- ToMediaType() : CoreGraphics::ImageFileFormat
- ToName() : Models::ModelNodeType
- ToString() : CoreAnimation::InfinityType, Script::InfoLog, CoreAnimation::SampleType, CoreGraphics::Adapter, Input::MouseButton, Animation::AnimJobEnqueueMode, CoreGraphics::AntiAliasQuality, CoreGraphics::IndexType, CoreGraphics::PixelFormat, CoreGraphics::PrimitiveTopology, Frame::LightingMode, Frame::SortingMode, CoreGraphics::BatchType, Lighting::LightType, Http::HttpMethod, Util::FourCC, CoreGraphics::ImageFileFormat, Http::HttpStatus, Input::Key, CoreAnimation::CurveType
- ToUpper() : Util::String
- TPJob() : Jobs::TPJob
- **TPJobCommand()**: Jobs::TPJobCommand
- **TPJobFuncDesc()**: Jobs::TPJobFuncDesc
- **TPJobPort()**: Jobs::TPJobPort
- **TPJobSlice()**: Jobs::TPJobSlice
- **TPJobSystem()**: Jobs::TPJobSystem
- **TPJobThreadPool()**: Jobs::TPJobThreadPool
- **TPWorkerThread()**: Jobs::TPWorkerThread
- **transform()**: Math::matrix44, Math::bbox
- **transform44()**: Math::transform44
- **TransformableProperty()**: BaseGameFeature::TransformableProperty
- **transformation()**: Math::matrix44
- **TransformDevice()**: CoreGraphics::TransformDevice
- **TransformDeviceBase()**: Base::TransformDeviceBase
- **TransformNode()**: Models::TransformNode
- **TransformNodeInstance()**: Models::TransformNodeInstance
- **Transition()**: FSM::Transition
- **translate()**: Math::matrix44
- **TranslateKeyCode()**: Win32::Win32DisplayDevice
- **translation()**: Math::matrix44
- **transpose()**: Math::matrix44
- **Trigger()**: Actions::SequenceAction, Core::CoreServer, Actions::Action, Actions::IfThenElseAction, Actions::FSMAction, Actions::Action
- **TriggerAction()**: PhysicsFeature::TriggerProperty
- **TriggerProperty()**: PhysicsFeature::TriggerProperty
- **Trim()**: Util::Blob, Util::String
- **TrimLeft()**: Util::String
- **TrimRight()**: Util::String
- **Type**: Base::TextureBase, Base::ShaderVariableBase, Base::TextureBase, Base::ShaderVariableBase
- **TypeToString()**: Util::Variant, Base::ShaderVariableBase, Input::InputEvent, Base::ShaderVariableBase

The Nebula Device 3 documentation generated by doxygen at Fri Mar
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- u -

- undenormalize() : Math::quaternion
- Unload() : CoreAnimation::AnimResource, Models::Model, Resources::Resource, Win360::D3D9IndexBuffer, Base::MeshBase, Resources::Resource, Direct3D9::D3D9Shader, Resources::Resource, Direct3D9::D3D9Texture, Base::MeshBase, Resources::Resource, Win360::D3D9VertexBuffer, Resources::ResourceDictionary, Base::VertexBufferBase, Direct3D9::D3D9Shader, Direct3D9::D3D9Texture, Win360::D3D9IndexBuffer, Win360::D3D9VertexBuffer
- UnloadCanScript() : Script::ScriptManager
- UnloadDialog() : Script::DialogManager
- UnloadOnScript() : Script::ScriptManager
- UnloadResources() : Characters::CharacterNode, Models::Model, Models::ModelNode, Models::ShapeNode, Models::StateNode, Particles::ParticleSystemNode, Models::ModelNode, Models::ShapeNode
- UnloadStateMachines() : Script::ScriptManager
- UnloadSubTasks() : Script::Task
- Unlock() : Resources::Resource
- UnlockSurfaces() : Resources::D3D9TextureStreamer
- UnlockTask() : Script::Task
- Unmap() : Win360::D3D9VertexBuffer
CoreAnimation::AnimKeyBuffer, IO::ZipFileStream, IO::MemoryStream, Direct3D9::D3D9Texture, Base::IndexBufferBase, Base::TextureBase, Win360::D3D9VertexBuffer, Base::VertexBufferBase, IO::Stream, Direct3D9::D3D9Texture, Win360::D3D9IndexBuffer, Win360::D3D9VertexBuffer, IO::FileStream

- UnmapCubeFace(): Direct3D9::D3D9Texture, Base::TextureBase, Direct3D9::D3D9Texture
- UnmapSampleCounts(): CoreAnimation::AnimSampleBuffer
- UnmapSamples(): CoreAnimation::AnimSampleBuffer
- UnmountArchive(): IO::IoServer
- UnmountStandardArchives(): IO::IoServer
- unpack_w(): Math::float4
- unpack_x(): Math::float4
- unpack_y(): Math::float4
- unpack_z(): Math::float4
- UnregisterAnimEventHandler(): Animation::AnimEventServer
- UnregisterDebugCounter(): Debug::DebugServer
- UnregisterDebugTimer(): Debug::DebugServer
- UnregisterEntity(): Visibility::VisibilityChecker
- UnregisterRenderModule(): Graphics::GraphicsServer
- UnregisterRTPlugin(): RenderModules::RTPluginRegistry
- UnregisterUnmanagedResource(): Resources::ResourceManager
- UnregisterUriScheme(): IO::SchemeRegistry
- upcast(): Ptr< TYPE >
- Update(): IO::ConsoleHandler, Jobs::JobDataDesc, RenderUtil::MayaCameraUtil, Jobs::JobUniformDesc, IO::ConsoleHandler, IO::Console, Resources::ResourceManager, IO::ConsoleHandler, GraphicsFeature::LightFlickerUtil, IO::ConsoleHandler, BaseGameFeature::TimeManager, IO::ConsoleHandler, FSM::StateMachine
- UpdateAudioListenerPosition(): GraphicsFeature::CameraProperty
- UpdateBoundingBox(): Models::Model
- UpdateButtonState(): XInput::XInputGamePad
- UpdateCamera(): GraphicsFeature::ChaseCameraProperty
- UpdateCameraLinks() : InternalGraphics::InternalStage
- UpdateCharacterSkins() : Characters::CharacterServer
- UpdateClipStatus() : InternalGraphics::InternalGraphicsEntity
- UpdateDialogData() : Script::Dialog
- UpdateGlobalBoundingBox() : InternalGraphics::InternalGraphicsEntity
- UpdateLightLinks() : InternalGraphics::InternalStage
- UpdateManagedTextureVariables() : Models::StateNode
- UpdateModelNodeInstanceShaderVariable() : Graphics::ModelEntity
- UpdateModelNodeInstanceVisibility() : Graphics::ModelEntity
- UpdatePointers() : Base::MouseRenderDeviceBase, Graphics::MouseRenderer
- UpdateProgressDisplay() : BaseGameFeature::LoaderServer
- UpdateRenderStats() : Resources::ManagedResource, Models::ModelInstance, Resources::ManagedResource
- UpdateShadowBuffers() : Lighting::SM30ShadowServer
- UpdateShadowTransforms() : Lighting::InternalAbstractLightEntity
- UpdateSoftwareSkinnedMeshes() : Base::SkinnedMeshRendererBase
- UpdateThumbAxis() : XInput::XInputGamePad
- UpdateTime() : BaseGameFeature::TimeSource, InternalGraphics::InternalGraphicsEntity, BaseGameFeature::TimeSource, Animation::AnimSequencer, InternalGraphics::InternalGraphicsEntity, BaseGameFeature::TimeSource
- UpdateTimePolling() : FrameSync::FrameSyncTimer
- UpdateTimes() : Animation::AnimJob
- UpdateTransform() : GraphicsFeature::GraphicsProperty
- UpdateTransforms() : StateObjectFeature::StateGraphicsProperty
- UpdateTriggerAxis() : XInput::XInputGamePad
- UpdateTriggeredEntities() : BaseGameFeature::EntityManager
- UpdateVertexStreams() : Direct3D9::D3D9ParticleSystemInstance
- UpdateViewMatrix() : Shared::CameraSettings
- UpdateVisibilityContext() : Visibility::VisibilityBoxSystem, Visibility::VisibilitySystemBase, Visibility::VisibilityChecker,
Visibility::VisibilityClusterSystem ,
Visibility::VisibilityQuadtree
- UpdateVisibilityLinks() : InternalGraphics::InternalView
- upvec() : Math::vector
- URI() : IO::URI
- Usage : Base::ResourceBase
- UserInfo() : IO::URI
- UserProfile() : BaseGameFeature::UserProfile
- UTF8ToWide() : Win32::Win32StringConverter
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Enumerator
- Related Functions

- _
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **v** -
  - Validate() : Threading::ObjectRef
  - ValidateDialog() : Script::DialogManager
  - ValidateModelInstance() : InternalGraphics::InternalModelEntity
  - Value() : Util::KeyValuePair< KEYTYPE, VALUETYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::KeyValuePair< KEYTYPE, VALUETYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::StringAtom
  - ValueAtIndex() : Util::Dictionary< KEYTYPE, VALUETYPE >
  - Values() : BaseGameFeature::CategoryManager::Entry
  - ValuesAs() : Util::Dictionary< KEYTYPE, VALUETYPE >
  - ValuesAsArray() : Util::Dictionary< KEYTYPE, VALUETYPE >
  - Variant() : Util::Variant
  - VariationLibrary() : Characters::Character
  - vec() : Math::line
  - vector() : Math::vector
  - Version() : Script::ActionReader
  - VertexBufferBase() : Base::VertexBufferBase
  - VertexComponent() : CoreGraphics::VertexComponent
  - VertexLayoutBase() : Base::VertexLayoutBase
  - VertexLayoutServerBase() : Base::VertexLayoutServerBase
  - View() : Graphics::View
  - ViewerApplication() : App::ViewerApplication
- VisibilityBoxSystem() : \texttt{Visibility::VisibilityBoxSystem}
- VisibilityChecker() : \texttt{Visibility::VisibilityChecker}
- VisibilityClusterSystem() : \texttt{Visibility::VisibilityClusterSystem}
- VisibilityContainer() : \texttt{Visibility::VisibilityContainer}
- VisibilityContext() : \texttt{Visibility::VisibilityContext}
- VisibilityQuadtree() : \texttt{Visibility::VisibilityQuadtree}
- VisibilityQuery() : \texttt{Visibility::VisibilityQuery}
- VisibilitySystemBase() : \texttt{Visibility::VisibilitySystemBase}
- VisResolveContainer() : \texttt{Models::VisResolveContainer< TYPE >}
- VisResolver() : \texttt{Models::VisResolver}
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
  a  
  b  
  c  
  d  
  e  
  f  
  g  
  h  
  i  
  j  
  k  
  l  
  m  
  n  
  o  
  p  
  q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **W** -

  - \( w() \): Math::float4, Math::quaternion
  - Wait() : Threading::SafePriorityQueue< PRITYPE, TYPE >, Messaging::AsyncPort, Win360::Win360Event, Threading::SafeQueue< TYPE >, Win360::Win360Event, Win360::Win360ThreadBarrier, Messaging::AsyncPort
  - WaitDone() : Jobs::TPJobPort, Jobs::SerialJobPort, Jobs::TPJobPort, Base::JobPortBase
  - WaitForFinished() : Visibility::VisibilityQuery
  - WaitForHandlersOpened() : Messaging::HandlerThreadBase
  - WaitForMessage() : Messaging::HandlerThreadBase, FrameSync::FrameSyncHandlerThread
  - WaitForPendingResources() : Graphics::GraphicsHandler, Graphics::GraphicsInterface, Resources::ResourceManager
  - WaitTimeout() : Win360::Win360Event, Threading::SafeQueue< TYPE >, Win360::Win360Event
  - WaitUpdateDone() : Characters::CharacterInstance
  - Warning() : IO::Console, IO::HistoryConsoleHandler, IO::ConsoleHandler, OSX::OSXConsoleHandler, IO::LogFileConsoleHandler, Script::InfoLog, Win32::Win32ConsoleHandler, IO::Console
  - WeakPtr() : WeakPtr< TYPE >
  - Weekday : Base::CalendarTimeBase
  - WeekdayToString() : Base::CalendarTimeBase
- WheelBackward() : Base::MouseBase
- WheelForward() : Base::MouseBase
- WideToUTF8() : Win32::Win32StringConverter
- Width() : Util::FixedTable< TYPE >
- width() : Math::rectangle< TYPE >
- Win32ConsoleHandler() : Win32::Win32ConsoleHandler
- Win32DisplayDevice() : Win32::Win32DisplayDevice
- Win32Guid() : Win32::Win32Guid
- Win32InputServer() : Win32::Win32InputServer
- Win32SkinnedMeshRenderer() : Win32::Win32SkinnedMeshRenderer
- Win32SystemInfo() : Win32::Win32SystemInfo
- Win360CriticalSection() : Win360::Win360CriticalSection
- Win360Event() : Win360::Win360Event
- Win360FileTime() : Win360::Win360FileTime
- Win360Heap() : Win360::Win360Heap
- Win360IpAddress() : Win360::Win360IpAddress
- Win360MemoryPool() : Win360::Win360MemoryPool
- Win360Socket() : Win360::Win360Socket
- Win360Thread() : Win360::Win360Thread
- Win360ThreadBarrier() : Win360::Win360ThreadBarrier
- Win360Timer() : Win360::Win360Timer
- WinProc() : Win32::Win32DisplayDevice
- Write() : IO::Stream, Conditions::And, Net::DebugPacket, Actions::SequenceAction, IO::Stream, Actions::Action, IO::MemoryStream, OSX::OSXFSWrapper, IO::Stream, IO::FileStream, Actions::IfThenElseAction, IO::Stream, IO::TextWriter, Win360::Win360FSWrapper, Actions::Action, Conditions::Not, Actions::ActionList, Conditions::Condition, Conditions::Or, Conditions::Condition
- WriteBlob() : IO::BinaryWriter
- WriteBool() : IO::BinaryWriter
- WriteChar() : IO::BinaryWriter, IO::TextWriter
- WriteComment() : IO::XmlWriter
- WriteContent() : Http::SvgPageWriter, IO::XmlWriter
- WriteDouble() : IO::BinaryWriter
- WriteFloat() : IO::BinaryWriter
- WriteFloat2() : IO::BinaryWriter
- WriteFloat4() : IO::BinaryWriter
- WriteFloatAsNormalizedUByte2() : IO::BinaryWriter
- WriteFloatAsUnsignedNormalizedUByte2() : IO::BinaryWriter
- WriteFormatted() : IO::TextWriter
- WriteGuid() : IO::BinaryWriter
- WriteInt() : IO::BinaryWriter
- WriteLine() : IO::TextWriter
- WriteLines() : IO::TextWriter
- WriteMatrix44() : IO::BinaryWriter
- WriteMessage() : Messaging::MessageWriter
- WriteMiniDump() : Win32::Win32MiniDump
- WritePoint() : IO::BinaryWriter
- WritePoolsToXML() : Resources::PoolResourceMapper
- WriteRaw() : Net::DebugPacket
- WriteRawData() : IO::BinaryWriter
- WriteRequestHeader() : Http::HttpRequestWriter
- WriteResponse() : Http::HttpResponseWriter
- WriteShort() : IO::BinaryWriter
- WriteString() : System::Win32Registry, IO::BinaryWriter, IO::TextWriter
- WriteTexturePoolToXML() : Resources::PoolResourceMapper
- WriteUChar() : IO::BinaryWriter
- WriteUInt() : IO::BinaryWriter
- WriteUShort() : IO::BinaryWriter
- WriteVector() : IO::BinaryWriter
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **x** -

- x() : Math::float2, Math::float4, Math::float2, Math::float4, Math::quaternion, Math::float4, Math::quaternion, Math::float4
- XInputGamePad() : XInput::XInputGamePad
- XmlReader() : IO::XmlReader
- XmlWriter() : IO::XmlWriter

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- y -

- y() : Math::float2, Math::float4, Math::float2, Math::quaternion, Math::float4, Math::quaternion, Math::float4
- Year : Base::CalendarTimeBase
- YieldThread() : Win360::Win360Thread, OSX::OSXThread, Win360::Win360Thread
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_ 
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **z** -

  - z() : Math::float4, Math::quaternion, Math::float4, Math::quaternion, Math::float4
  - ZipArchive() : IO::ZipArchive
  - ZipDirEntry() : IO::ZipDirEntry
  - ZipFileEntry() : IO::ZipFileEntry
  - ZipFileStream() : IO::ZipFileStream
  - ZipFileSystem() : IO::ZipFileSystem
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Enumerator
- Related Functions

- _
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- ~AbstractLightEntity(): Graphics::AbstractLightEntity
- ~Action(): Actions::Action
- ~ActionReader(): Script::ActionReader
- ~ActorPhysicsProperty(): PhysicsFeature::ActorPhysicsProperty
- ~AnimEventHandlerBase(): Animation::AnimEventHandlerBase
- ~AnimEventManager(): BaseGameFeature::AnimEventManager
- ~AnimEventServer(): Animation::AnimEventServer
- ~AnimJob(): Animation::AnimJob
- ~AnimKeyBuffer(): CoreAnimation::AnimKeyBuffer
- ~AnimResource(): CoreAnimation::AnimResource
- ~AnimSampleBuffer(): CoreAnimation::AnimSampleBuffer
- ~AnimSequencer(): Animation::AnimSequencer
- ~Application(): App::Application
- ~ArchiveBase(): IO::ArchiveBase
- ~ArchiveFileSystem(): IO::ArchiveFileSystem
- ~Array(): Util::Array< TYPE >
- ~AssignRegistry(): IO::AssignRegistry
- ~AsyncPort(): Messaging::AsyncPort
- ~AttachmentManager(): GraphicsFeature::AttachmentManager
- ~AttachmentServer(): `InternalGraphics::AttachmentServer`
- ~BinaryReader(): `IO::BinaryReader`
- ~BinaryWriter(): `IO::BinaryWriter`
- ~Blob(): `Util::Blob`
- ~BXmILoaderUtil(): `IO::BXmILoaderUtil`
- ~BXmIReader(): `IO::BXmIReader`
- ~CameraEntity(): `Graphics::CameraEntity`
- ~CameraProperty(): `GraphicsFeature::CameraProperty`
- ~CategoryManager(): `BaseGameFeature::CategoryManager`
- ~Character(): `Characters::Character`
- ~CharacterAnimationController(): `Characters::CharacterAnimationController`
- ~CharacterInstance(): `Characters::CharacterInstance`
- ~CharacterJoint(): `Characters::CharacterJoint`
- ~CharacterNode(): `Characters::CharacterNode`
- ~CharacterNodeInstance(): `Characters::CharacterNodeInstance`
- ~CharacterServer(): `Characters::CharacterServer`
- ~CharacterSkeleton(): `Characters::CharacterSkeleton`
- ~CharacterSkeletonInstance(): `Characters::CharacterSkeletonInstance`
- ~CharacterSkin(): `Characters::CharacterSkin`
- ~CharacterSkinLibrary(): `Characters::CharacterSkinLibrary`
- ~CharacterSkinList(): `Characters::CharacterSkinList`
- ~CharacterSkinNode(): `Characters::CharacterSkinNode`
- ~CharacterSkinNodeInstance(): `Characters::CharacterSkinNodeInstance`
- ~CharacterSkinSet(): `Characters::CharacterSkinSet`
- ~CharacterVariationSet(): `Characters::CharacterVariationSet`
- ~ChaseCameraProperty(): `GraphicsFeature::ChaseCameraProperty`
- ~Console(): `IO::Console`
- ~ConsoleApplication(): `App::ConsoleApplication`
- ~ConsoleHandler(): `IO::ConsoleHandler`
- ~ConsolePageHandler(): `Debug::ConsolePageHandler`
- ~CoreServer(): `Core::CoreServer`
- ~CreateEntityCommand(): `Commands::CreateEntityCommand`
- ~D3D9DisplayDevice(): `Direct3D9::D3D9DisplayDevice`
- ~D3D9IndexBuffer(): `Win360::D3D9IndexBuffer`
- ~D3D9ParticleRenderer(): Direct3D9::D3D9ParticleRenderer
- ~D3D9ParticleSystemInstance(): Direct3D9::D3D9ParticleSystemInstance
- ~D3D9RenderDevice(): Direct3D9::D3D9RenderDevice
- ~D3D9RenderTarget(): Direct3D9::D3D9RenderTarget
- ~D3D9Shader(): Direct3D9::D3D9Shader
- ~D3D9ShaderInstance(): Direct3D9::D3D9ShaderInstance
- ~D3D9ShaderServer(): Direct3D9::D3D9ShaderServer
- ~D3D9ShaderVariable(): Direct3D9::D3D9ShaderVariable
- ~D3D9ShaderVariation(): Direct3D9::D3D9ShaderVariation
- ~D3D9ShapeRenderer(): Win360::D3D9ShapeRenderer
- ~D3D9TextRenderer(): Direct3D9::D3D9TextRenderer
- ~D3D9Texture(): Direct3D9::D3D9Texture
- ~D3D9TransformDevice(): Win360::D3D9TransformDevice
- ~D3D9VertexBuffer(): Win360::D3D9VertexBuffer
- ~D3D9VertexLayout(): Win360::D3D9VertexLayout
- ~DebugCounter(): Debug::DebugCounter
- ~DebugGraphicsHandler(): Debug::DebugGraphicsHandler
- ~DebugHandler(): Debug::DebugHandler
- ~DebugInterface(): Debug::DebugInterface
- ~DebugPacket(): Net::DebugPacket
- ~DebugServer(): Debug::DebugServer
- ~DebugShapeRenderer(): Debug::DebugShapeRenderer
- ~DebugTextRenderer(): Debug::DebugTextRenderer
- ~DebugTimer(): Debug::DebugTimer
- ~Dialog(): Script::Dialog
- ~DialogManager(): Script::DialogManager
- ~DialogTake(): Script::DialogTake
- ~Display(): Graphics::Display
- ~DisplayDevice(): CoreGraphics::DisplayDevice
- ~DisplayDeviceBase(): Base::DisplayDeviceBase
- ~DisplayEventHandler(): CoreGraphics::DisplayEventHandler
- ~DrawFullScreenQuad(): RenderUtil::DrawFullScreenQuad
- ~EmitterMesh(): Particles::EmitterMesh
- ~Entity(): Game::Entity
- ~EntityLoaderBase(): BaseGameFeature::EntityLoaderBase
- ~EntityManager(): BaseGameFeature::EntityManager
- ~EnvelopeSampleBuffer(): Particles::EnvelopeSampleBuffer
- ~EnvEntityManager(): BaseGameFeature::EnvEntityManager
- EnvironmentCollideProperty() : PhysicsFeature::EnvironmentCollideProperty
- EnvQueryManager() : BaseGameFeature::EnvQueryManager
- ExcelXmlReader() : IO::ExcelXmlReader
- ExitHandler() : Core::ExitHandler
- Extrapolator() : Math::Extrapolator<TYPE>
- FactoryManager() : BaseGameFeature::FactoryManager
- FeatureUnit() : Game::FeatureUnit
- FileStream() : IO::FileStream
- FixedArray() : Util::FixedArray<TYPE>
- FixedTable() : Util::FixedTable<TYPE>
- FocusManager() : BaseGameFeature::FocusManager
- FrameBatch() : Frame::FrameBatch
- FramePass() : Frame::FramePass
- FramePassBase() : Frame::FramePassBase
- FramePostEffect() : Frame::FramePostEffect
- FrameServer() : Frame::FrameServer
- FrameShader() : Frame::FrameShader
- FrameSyncHandlerThread() : FrameSync::FrameSyncHandlerThread
- FrameSyncSharedData() : FrameSync::FrameSyncSharedData
- FrameSyncTimer() : FrameSync::FrameSyncTimer
- GameApplication() : App::GameApplication
- GameContentServer() : IO::GameContentServer
- GameContentServerBase() : Base::GameContentServerBase
- GamePadBase() : Base::GamePadBase
- GameServer() : Game::GameServer
- GameStateHandler() : BaseGameFeature::GameStateHandler
- GlobalAttrsManager() : BaseGameFeature::GlobalAttrsManager
- GlobalStringAtomTable() : Util::GlobalStringAtomTable
- GraphicsEntity() : Graphics::GraphicsEntity
- GraphicsHandler() : Graphics::GraphicsHandler
- GraphicsInterface() : Graphics::GraphicsInterface
- GraphicsProperty() : GraphicsFeature::GraphicsProperty
- GraphicsServer() : Graphics::GraphicsServer
- Handler() : Messaging::Handler
- HtmlPageWriter() : Http::HtmlPageWriter
- ~HttpInterface() : Http::HttpInterface
- ~HttpMessageHandler() : Http::HttpMessageHandler
- ~HttpRequest() : Http::HttpRequest
- ~HttpRequestHandler() : Http::HttpRequestHandler
- ~HttpServer() : Http::HttpServer
- ~HttpServerProxy() : Http::HttpServerProxy
- ~IndexBufferBase() : Base::IndexBufferBase
- ~InfoLog() : Script::InfoLog
- ~InputHandler() : Input::InputHandler
- ~InputServer() : Input::InputServer
- ~InputServerBase() : Base::InputServerBase
- ~InputTimeSource() : BaseGameFeature::InputTimeSource
- ~InterfaceBase() : Interface::InterfaceBase
- ~InternalAbstractLightEntity() : Lighting::InternalAbstractLightEntity
- ~InternalCameraEntity() : InternalGraphics::InternalCameraEntity
- ~InternalModelEntity() : InternalGraphics::InternalModelEntity
- ~InternalStage() : InternalGraphics::InternalStage
- ~InternalView() : InternalGraphics::InternalView
- ~IoInterfaceHandler() : IO::IoInterfaceHandler
- ~IoServer() : IO::IoServer
- ~JobBase() : Base::JobBase
- ~JobPortBase() : Base::JobPortBase
- ~JobSystem() : Jobs::JobSystem
- ~KeyboardBase() : Base::KeyboardBase
- ~LightPrePassServer() : Lighting::LightPrePassServer
- ~LightServer() : Lighting::LightServer
- ~LightServerBase() : Lighting::LightServerBase
- ~List() : Util::List<TYPE>
- ~LoaderServer() : BaseGameFeature::LoaderServer
- ~LoadingResource() : Resources::LoadingResource
- ~LocalStringAtomTable() : Util::LocalStringAtomTable
- ~LogFileConsoleHandler() : IO::LogFileConsoleHandler
- ~ManagedResource() : Resources::ManagedResource
- ~Manager() : Game::Manager
- ~MayaCameraProperty() : GraphicsFeature::MayaCameraProperty
- ~MemoryStream() : IO::MemoryStream
- ~MeshBase() : Base::MeshBase
- ~MessageClient() : Net::MessageClient
- ~Model() : Models::Model
- ~ModelEntity() : Graphics::ModelEntity
- ~ModelInstance() : Models::ModelInstance
- ~ModelNode() : Models::ModelNode
- ~ModelNodeInstance() : Models::ModelNodeInstance
- ~ModelServer() : Models::ModelServer
- ~MouseBase() : Base::MouseBase
- ~MouseGripperProperty() : PhysicsFeature::MouseGripperProperty
- ~MouseRenderDevice() : CoreGraphics::MouseRenderDevice
- ~MouseRenderDeviceBase() : Base::MouseRenderDeviceBase
- ~MouseRenderer() : Graphics::MouseRenderer
- ~MultipleRenderTargetBase() : Base::MultipleRenderTargetBase
- ~Node() : Util::SimpleTree< VALUETYPE >::Node, Util::QuadTree< TYPE >::Node
- ~ObjectRef() : Threading::ObjectRef
- ~ObserverContext() : Visibility::ObserverContext
- ~OSXCriticalSection() : OSX::OSXCriticalSection
- ~OSXHeap() : OSX::OSXHeap
- ~OSXMemoryPool() : OSX::OSXMemoryPool
- ~OSXThread() : OSX::OSXThread
- ~OSXThreadLocalPtr() : OSX::OSXThreadLocalPtr< TYPE >
- ~ParticleRenderer() : Particles::ParticleRenderer
- ~ParticleRendererBase() : Base::ParticleRendererBase
- ~ParticleServer() : Particles::ParticleServer
- ~ParticleSystem() : Particles::ParticleSystem
- ~ParticleSystemNode() : Particles::ParticleSystemNode
- ~ParticleSystemNodeInstance() : Particles::ParticleSystemNodeInstance
- ~PhysicsProperty() : PhysicsFeature::PhysicsProperty
- ~PlayClipJob() : Animation::PlayClipJob
- ~PoolArrayAllocator() : Memory::PoolArrayAllocator
- ~PoolLoadingResource() : Resources::PoolLoadingResource
- ~PoolResourceMapper() : Resources::PoolResourceMapper
- ~Property() : Game::Property
- ~Ptr() : Ptr<TYPE>
- ~RefCounted() : Core::RefCounted
- ~RenderApplication() : App::RenderApplication
- ~RenderDevice() : CoreGraphics::RenderDevice
- ~RenderWindowDeviceBase() : Base::RenderWindowDeviceBase
- ~RenderEventHandler() : CoreGraphics::RenderEventHandler
- ~RenderModule() : RenderModules::RenderModule
- ~RenderTargetBase() : Base::RenderTargetBase
- ~Resource() : Resources::Resource
- ~ResourceBase() : Base::ResourceBase
- ~ResourceDictionary() : Resources::ResourceDictionary
- ~ResourceLoader() : Resources::ResourceLoader
- ~ResourceManager() : Resources::ResourceManager
- ~ResourceMapper() : Resources::ResourceMapper
- ~ResourceSaver() : Resources::ResourceSaver
- ~RingBuffer() : Util::RingBuffer<TYPE>
- ~RTPlugin() : RenderModules::RTPlugin
- ~RTPluginRegistry() : RenderModules::RTPluginRegistry
- ~SchemeRegistry() : IO::SchemeRegistry
- ~ScriptFeatureUnit() : ScriptFeature::ScriptFeatureUnit
- ~ScriptManager() : Script::ScriptManager
- ~SequenceAction() : Actions::SequenceAction
- ~SerialJob() : Jobs::SerialJob
- ~SerialJobPort() : Jobs::SerialJobPort
- ~SerialJobSystem() : Jobs::SerialJobSystem
- ~ShaderBase() : Base::ShaderBase
- ~ShaderInstanceBase() : Base::ShaderInstanceBase
- ~ShaderServer() : CoreGraphics::ShaderServer
- ~ShaderServerBase() : Base::ShaderServerBase
- ~ShaderVariableBase() : Base::ShaderVariableBase
- ~ShaderVariableInstanceBase() : Base::ShaderVariableInstanceBase
- ~ShaderVariationBase() : Base::ShaderVariationBase
- ~ShadowServer() : Lighting::ShadowServer
- ~ShapeNode() : Models::ShapeNode
- ~ShapeNodeInstance() : Models::ShapeNodeInstance
- ~ShapeRenderer() : CoreGraphics::ShapeRenderer
- ~ShapeRendererBase() : Base::ShapeRendererBase
- ~SimpleResourceMapper() : Resources::SimpleResourceMapper
- ~SkinnedMeshRenderer() : Characters::SkinnedMeshRenderer
- ~SkinnedMeshRendererBase() : Base::SkinnedMeshRendererBase
- ~SM30LightServer() : Lighting::SM30LightServer
- ~SM30ShadowServer() : Lighting::SM30ShadowServer
- ~State() : FSM::State
- ~StateGraphicsProperty() : StateObjectFeature::StateGraphicsProperty
- ~StateMachine() : FSM::StateMachine
- ~StateNode() : Models::StateNode
- ~StateNodeInstance() : Models::StateNodeInstance
- ~StdTcpClient() : Net::StdTcpClient
- ~StdTcpClientConnection() : Net::StdTcpClientConnection
- ~StdTcpServer() : Net::StdTcpServer
- ~Stream() : IO::Stream
- ~StreamModelLoader() : Models::StreamModelLoader
- ~StreamReader() : IO::StreamReader
- ~StreamResourceLoader() : Resources::StreamResourceLoader
- ~StreamTextureSaverBase() : Base::StreamTextureSaverBase
- ~StreamWriter() : IO::StreamWriter
- ~String() : Util::String
- ~StringAtomTableBase() : Util::StringAtomTableBase
- ~StringBuffer() : Util::StringBuffer
- ~SubstitutionManager() : Script::SubstitutionManager
- ~SvgLineChartWriter() : Http::SvgLineChartWriter
- ~SvgPageWriter() : Http::SvgPageWriter
- ~SystemTimeSource() : BaseGameFeature::SystemTimeSource
- ~Task() : Script::Task
- ~TcpMessageCodec() : Net::TcpMessageCodec
- ~TextReader() : IO::TextReader
- ~TextRenderer() : CoreGraphics::TextRenderer
- ~TextRendererBase() : Base::TextRendererBase
- ~TextureBase() : Base::TextureBase
- ~TexturePoolMapperScheduler() :
  Resources::TexturePoolMapperScheduler
- ~TextWriter() : IO::TextWriter
- ~ThreadSafeDisplayEventHandler() :
  CoreGraphics::ThreadSafeDisplayEventHandler
- ~ThreadSafeRenderEventHandler() :
  CoreGraphics::ThreadSafeRenderEventHandler
- ~TimeManager() : BaseGameFeature::TimeManager
- ~TimeSource() : BaseGameFeature::TimeSource
- ~TPJob() : Jobs::TPJob
- ~TPJobPort() : Jobs::TPJobPort
- ~TPJobSlice() : Jobs::TPJobSlice
- ~TPJobSystem() : Jobs::TPJobSystem
- ~TPJobThreadPool() : Jobs::TPJobThreadPool
- ~TPWorkerThread() : Jobs::TPWorkerThread
- ~TransformableProperty() :
  BaseGameFeature::TransformableProperty
- ~TransformDevice() : CoreGraphics::TransformDevice
- ~TransformDeviceBase() : Base::TransformDeviceBase
- ~TransformNode() : Models::TransformNode
- ~TransformNodeInstance() : Models::TransformNodeInstance
- ~Transition() : FSM::Transition
- ~TriggerProperty() : PhysicsFeature::TriggerProperty
- ~UserProfile() : BaseGameFeature::UserProfile
- ~Variant() : Util::Variant
- ~VertexBufferBase() : Base::VertexBufferBase
- ~VertexLayoutBase() : Base::VertexLayoutBase
- ~VertexLayoutServerBase() : Base::VertexLayoutServerBase
- ~View() : Graphics::View
- ~ViewerApplication() : App::ViewerApplication
- ~VisibilityBoxSystem() : Visibility::VisibilityBoxSystem
- ~VisibilityChecker() : Visibility::VisibilityChecker
- ~VisibilityClusterSystem() : Visibility::VisibilityClusterSystem
- ~VisibilityContainer() : Visibility::VisibilityContainer
- ~VisibilityContext() : Visibility::VisibilityContext
- ~VisibilityQuadtree() : Visibility::VisibilityQuadtree
- ~VisibilityQuery() : Visibility::VisibilityQuery
- ~VisibilitySystemBase() : Visibility::VisibilitySystemBase
- ~VisResolver() : Models::VisResolver
- ~WeakPtr() : WeakPtr< TYPE >
- ~Win32DisplayDevice() : Win32::Win32DisplayDevice
- ~Win32InputServer() : Win32::Win32InputServer
- ~Win32SkinnedMeshRenderer() : Win32::Win32SkinnedMeshRenderer
- ~Win360CriticalSection() : Win360::Win360CriticalSection
- ~Win360Event() : Win360::Win360Event
- ~Win360Heap() : Win360::Win360Heap
- ~Win360MemoryPool() : Win360::Win360MemoryPool
- ~Win360Socket() : Win360::Win360Socket
- ~Win360Thread() : Win360::Win360Thread
- ~Win360ThreadBarrier() : Win360::Win360ThreadBarrier
- ~XInputGamePad() : XInput::XInputGamePad
- ~XmlReader() : IO::XmlReader
- ~XmlWriter() : IO::XmlWriter
- ~ZipArchive() : IO::ZipArchive
- ~ZipFileEntry() : IO::ZipFileEntry
- ~ZipFileStream() : IO::ZipFileStream
- ~ZipFileSystem() : IO::ZipFileSystem
Namespace Members

- __Dispatcher() : Messaging
- __Handler() : Messaging
- __HeapAlignPointerAndWritePadding16() : Memory
- __HeapAlloc16() : Memory
- __HeapFree16() : Memory
- __HeapReAlloc16() : Memory
- a -
  - Alloc() : Memory
  - AnimJobUtilMix() : CoreAnimation
  - AnimJobUtilSampleLinear() : CoreAnimation
  - AnimJobUtilSampleStep() : CoreAnimation

- c -
  - Clear() : Memory
  - Copy() : Memory
  - CopyToGraphicsMemory() : Memory

- d -
  - DumpTotalMemoryStatus() : Memory
  - DuplicateCString() : Memory

- e -
  - Exit() : Application

- f -
  - Fill() : Memory
  - Free() : Memory

- g -
  - GetTotalMemoryStatus() : Memory

- i -
  - IsOverlapping() : Memory
- n -
  - n_angulardistance() : Math
  - n_clamp() : Math
  - n_fchop() : Math
  - n_fequal() : Math
  - n_fgreater() : Math
  - n_fless() : Math
  - n_frnd() : Math
  - n_iclamp() : Math
  - n_isdenormal() : Math
  - n_lerp() : Math
  - n_log() : Math
  - n_log2() : Math
  - n_modangle() : Math
  - n_nearequal() : Math
  - n_rand() : Math
  - n_saturate() : Math
  - n_undenormalize() : Math

- r -
  - Realloc() : Memory

- s -
  - SecondsToTicks() : Timing
  - SetupHeaps() : Memory
  - Sleep() : Timing

- t -
  - TicksToSeconds() : Timing
- __HeapAlignPointerAndWritePadding16() : win360memoryconfig.h
- __HeapAlloc16() : win360memoryconfig.h
- __HeapFree16() : win360memoryconfig.h
- __HeapReAlloc16() : win360memoryconfig.h
- __HeapSize16() : win360memoryconfig.h
- __HeapUnalignPointer16() : win360memoryconfig.h
- AnimJobUtilMix() : animjobutil.h
- AnimJobUtilSampleLinear() : animjobutil.h
- AnimJobUtilSampleStep() : animjobutil.h
- CopyToGraphicsMemory() : win32memory.h
- n_angulardistance() : scalar.h
- n_clamp() : scalar.h
- n_fequal() : scalar.h
- n_fgreater() : scalar.h
- n_fless() : scalar.h
- n_iclamp() : scalar.h
- n_isdenormal() : scalar.h
- n_lerp() : scalar.h
- n_nearequal() : scalar.h
- n_saturate() : scalar.h
- n_undenormalize() : scalar.h
- operator new() : osxmemory.h
- SecondsToTicks() : time.h
- Sleep() : time.h
- TicksToSeconds() : time.h
Heaps : osxmemoryconfig.h, win360memoryconfig.h
JobMaxSliceSize : config.h
The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:51 2010
- a -

- a() : Math::plane
- abs() : Math::float2, Math::float4
- AbstractLightEntity() : Graphics::AbstractLightEntity
- Accept() : Win360::Win360Socket
- AcceptsMessage() : Messaging::Port, Game::Entity, Messaging::Port
- AcquireJointTextureRowPointer() : Base::SkinnedMeshRendererBase
- Action() : Actions::Action
- ActionReader() : Script::ActionReader
- ActivateDialog() : Script::Dialog, Script::DialogManager
- ActivateEntity() : BaseGameFeature::EntityManager
- ActorPhysicsProperty() : PhysicsFeature::ActorPhysicsProperty
- AdapterExists() : Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice, Graphics::Display, Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice
- AdapterInfo() : CoreGraphics::AdapterInfo
- Add() : Util::Dictionary< KEYTYPE, VALUETYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::RingBuffer< TYPE >, Models::VisResolveContainer< TYPE >, OSX::OSXInterlocked, Win360::Win360Interlocked
- AddAction() : Script::DialogTake, FSM::Transition
- AddAfter() : Util::List< TYPE >
- `AddAttr()` : `Http::HtmlPageWriter`
- `AddBack()` : `Util::List<TYPE>`
- `AddBatch()` : `Frame::FramePassBase`
- `AddBefore()` : `Util::List<TYPE>`
- `AddBlob()` : `Game::Entity`
- `AddBool()` : `Game::Entity`
- `AddCategoryAttr()` : `BaseGameFeature::CategoryManager`
- `AddChild()` : `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`
- `AddChildTake()` : `Script::DialogTake`
- `AddCondition()` : `Conditions::And`, `Conditions::Or`, `FSM::Transition`
- `AddCurParticleVertexIndex()` : `Direct3D9::D3D9ParticleRenderer`
- `AddDeferredMessage()` : `InternalGraphics::InternalGraphicsEntity`
- `AddDelayedJob()` : `BaseGameFeature::EntityManager`
- `AddDependency()` : `InternalGraphics::InternalView`
- `AddDepthStencilBuffer()` : `Base::RenderTargetBase`
- `AddEntryAction()` : `FSM::State`
- `AddEvent()` : `CoreAnimation::AnimClip`
- `AddExitAction()` : `FSM::State`
- `AddFloat()` : `Game::Entity`
- `AddFloat4()` : `Game::Entity`
- `AddFragment()` : `Characters::CharacterSkinNode`
- `AddFrameAction()` : `FSM::State`
- `AddFramePassBase()` : `Frame::FrameShader`
- `AddFront()` : `Util::List<TYPE>`
- `AddGraphicsAttachment()` : `GraphicsFeature::AttachmentManager`
- `AddGraphicsAttachmentTemporary()` : `GraphicsFeature::AttachmentManager`
- `AddGuid()` : `Game::Entity`
- `AddInfo()` : `Script::InfoLog`
- `AddInt()` : `Game::Entity`
- `AddLink()` : `InternalGraphics::InternalGraphicsEntity`
- AddList() : Util::List< TYPE >
- AddMatrix44() : Game::Entity
- AddMessage() : Messaging::BatchMessage, Messaging::BlockingHandlerThread, FrameSync::FrameSyncHandlerThread, Messaging::HandlerThreadBase, Messaging::RunThroughHandlerThread
- AddMultiple() : Util::SparseTable< TYPE >
- AddMultipleRenderTarget() : Frame::FrameShader
- AddProperties() : BaseGameFeature::FactoryManager
- AddRef() : Core::RefCounted, Util::SimpleTree< VALUETYPE >::Node, Core::RefCounted
- AddReference() : Util::SparseTable< TYPE >
- AddRenderTarget() : Base::MultipleRenderTargetBase, Frame::FrameShader, Base::MultipleRenderTargetBase
- AddSample() : Math::Extrapolator< TYPE >
- AddShaderParam() : Models::StateNode
- AddShape() : Base::ShapeRendererBase
- AddShapes() : Base::ShapeRendererBase, PhysicsFeature::EnvironmentCollideProperty
- AddSharedDepthStencilBuffer() : Base::RenderTargetBase
- AddSingle() : Util::SparseTable< TYPE >
- AddSkin() : Characters::CharacterSkinSet, Characters::CharacterSkinLibrary
- AddSkinList() : Characters::CharacterSkinLibrary
- AddState() : FSM::StateMachine
- AddStateHandler() : App::GameApplication
- AddString() : Util::StringBuffer, Game::Entity
- AddText() : Script::DialogTake
- AddTextElement() : Base::TextRendererBase
- AddTextElements() : Base::TextRendererBase
- AddTexture() : Frame::FrameShader
- AddTrack() : Http::SvgLineChartWriter
- AddTrackedCharJoint() : InternalGraphics::InternalModelEntity, Graphics::ModelEntity
- AddTransition() : FSM::State
- AddVariable() : Frame::FramePassBase, Frame::FrameBatch, Frame::FramePassBase
- AddVisibleNodeInstance() : Models::ModelNode
- `AddVisibleParticleSystem()` : `Base::ParticleRendererBase`
- `AddWireFrameBox()` : `Base::ShapeRendererBase`
- `AdvanceProgress()` : `BaseGameFeature::LoaderServer`
- `affinetransformation()` : `Math::matrix44`
- `all()` : `Math::float2`
- `Alloc()` : `OSX::OSXHeap`, `OSX::OSXMemoryPool`, `Memory::PoolArrayAllocator`, `Win360::Win360MemoryPool`, `Win360::Win360Heap`
- `AllocInstanceMemory()` : `Core::Rtti`
- `AllocJointTextureRow()` : `Base::SkinnedMeshRendererBase`
- `AllocPrivateBuffer()` : `Base::JobBase`
- `And()` : `Util::BitField< NUMBITS >`
- `angle()` : `Math::float4`
- `AnimationLibrary()` : `Characters::Character`
- `AnimClip()` : `CoreAnimation::AnimClip`
- `AnimController()` : `Characters::CharacterInstance`
- `AnimCurve()` : `CoreAnimation::AnimCurve`
- `AnimEvent()` : `CoreAnimation::AnimEvent`
- `AnimEventHandlerBase()` : `Animation::AnimEventHandlerBase`
- `AnimEventInfo()` : `Animation::AnimEventInfo`
- `AnimEventManager()` : `BaseGameFeature::AnimEventManager`
- `AnimEventServer()` : `Animation::AnimEventServer`
- `AnimJob()` : `Animation::AnimJob`
- `AnimKeyBuffer()` : `CoreAnimation::AnimKeyBuffer`
- `AnimResource()` : `CoreAnimation::AnimResource`
- `AnimSampleBuffer()` : `CoreAnimation::AnimSampleBuffer`
- `AnimSequencer()` : `Animation::AnimSequencer`, `Characters::CharacterAnimationController`
- `any()` : `Math::float2`
- `Append()` : `Util::String`, `Util::SimpleTree< VALUETYPE >::Node`, `Util::String`, `Util::Array< TYPE >`
- `AppendAction()` : `Actions::SequenceAction`
- `AppendArray()` : `Util::Array< TYPE >`
- `AppendBool()` : `Util::String`
- `AppendCommandString()` : `Util::CommandLineArgs`
- `AppendFloat()` : `Util::String`
- `AppendFloat2()` : `Util::String`
- `AppendFloat4()` : `Util::String`
- `AppendInt()` : `Util::String`
- AppendLoadingResource() : Resources::ResourceScheduler
- AppendLocalPath() : IO::URI
- AppendMatrix44() : Util::String
- AppendRange() : Util::String
- Application() : App::Application
- Apply() : Base::ShaderVariableInstanceBase
- ApplyCameraSettings() : InternalGraphics::InternalView
- ApplyGlobalAudioPitch() : BaseGameFeature::TimeManager
- ApplyImpulseAtPos() : PhysicsFeature::PhysicsProperty
- ApplyJointComponents() : Characters::CharacterSkeletonInstance
- ApplyModelEntityLights() : Lighting::LightServerBase, Lighting::SM30LightServer, Lighting::LightServerBase
- ApplyModelTransforms() : Win360::D3D9TransformDeviceBase, Win360::D3D9TransformDevice
- ApplyObjectId() : Base::ShaderServerBase
- ApplyPrimitives() : Base::MeshBase
- ApplySharedState() : Models::StateNode, Models::ShapeNode, Models::ModelNode, Characters::CharacterSkinNode, Models::ModelNode, Models::StateNode, Models::ModelNode
- ApplySkinList() : Characters::CharacterSkinSet
- ApplyState() : Models::ModelNodeInstance, Models::TransformNodeInstance, Models::ModelNodeInstance, Models::StateNodeInstance, Models::StateNodeInstance
- ApplyTimeEffect() : BaseGameFeature::TimeManager
- ApplyViewSettings() : Base::TransformDeviceBase, Win360::D3D9TransformDevice
- ArchiveBase() : IO::ArchiveBase
- ArchiveFileSystem() : IO::ArchiveFileSystem
- AreKeySliceValuesValid() : CoreAnimation::AnimClip
- AreMipMapsEnabled() : Base::RenderTargetBase
- AreSubTasksLoaded() : Script::Task
- Array() : Util::Array<TYPE>
- Arrive() : Win360::Win360ThreadBarrier
- ArriveAtSyncPoint() : FrameSync::FrameSyncHandlerThread
- As() : Util::Array<TYPE>
- `as()` : `Math::frustum`
- `As()` : `Util::String`
- `as()` : `Math::bbox`, `Math::matrix44`, `Math::float2`
- `AsStringArray()` : `Util::RingBuffer< TYPE >`, `Util::FixedArray< TYPE >`
- `AsBinary()` : `Win32::Win32Guid`, `OSX::OSXGuid`
- `AsBool()` : `Util::String`
- `AsCharPtr()` : `Util::String`
- `AsD3D9MultiSampleType()` : `Win360::D3D9Types`
- `AsD3D9PixelFormat()` : `Win360::D3D9Types`
- `AsD3D9Pool()` : `Win360::D3D9Types`
- `AsD3D9PrimitiveType()` : `Win360::D3D9Types`
- `AsD3D9Usage()` : `Win360::D3D9Types`
- `AsD3D9VertexDeclarationType()` : `Win360::D3D9Types`
- `AsD3D9VertexDeclarationUsage()` : `Win360::D3D9Types`
- `AsD3DXImageFileFormat()` : `Win360::D3D9Types`
- `AsFloat()` : `Util::String`
- `AsFloat2()` : `Util::String`
- `AsFloat4()` : `Util::String`
- `AsInt()` : `Util::String`
- `AsMatrix44()` : `Util::String`
- `AsNebulaPixelFormat()` : `Win360::D3D9Types`
- `AsRootKey()` : `System::Win32Registry`
- `AsSampleIndex()` : `Particles::EnvelopeSampleBuffer`
- `Assert()` : `Actions::ActionList`, `Conditions::Not`,
  `Actions::SequenceAction`, `Conditions::Condition`,
  `Actions::Action`, `Script::DialogTake`, `Conditions::Not`,
  `Conditions::And`, `Actions::Action`, `Conditions::Condition`,
  `Actions::ActionList`, `Actions::Action`, `Conditions::Or`,
  `Conditions::And`, `Actions::IfThenElseAction`, `Conditions::Or`,
  `Script::Task`
- `AssertDialog()` : `Script::Dialog`
- `AssertDialogTakeExists()` : `Script::DialogManager`
- `Assign()` : `IO::Assign`
- `AssignRegistry()` : `IO::AssignRegistry`
- `AssignRenderBufferTextures()` : `Lighting::LightPrePassServer`
- `AsString()` : `Win32::Win32Guid`, `IO::URI`,
  `Win360::Win360FileTime`, `OSX::OSXGuid`, `Util::StringAtom`,
  `Util::FourCC`, `IO::MediaType`
- AsUInt() : Util::FourCC
- AsyncPort() : Messaging::AsyncPort
- At() : Util::FixedTable< TYPE >
- AttachDisplayEventHandler() : Graphics::Display
- AttachEntity() : BaseGameFeature::EntityManager, InternalGraphics::AttachmentServer, InternalGraphics::InternalStage
- AttachEntityLoader() : BaseGameFeature::LoaderServer
- AttachEntityTemporary() : InternalGraphics::AttachmentServer
- AttachEventHandler() : Base::RenderDeviceBase, Base::DisplayDeviceBase, Base::RenderDeviceBase, Base::DisplayDeviceBase
- AttachGameFeature() : Game::GameServer
- AttachHandler() : Messaging::Port, IO::Console, Messaging::Port, Interface::InterfaceBase, Messaging::Port, Messaging::AsyncPort, Messaging::Port, Interface::InterfaceBase, Messaging::Port, Messaging::HandlerThreadBase, Messaging::Port, Messaging::HandlerThreadBase, Messaging::Port, Messaging::HandlerThreadBase, Interface::InterfaceBase, Messaging::HandlerThreadBase, Interface::InterfaceBase
- AttachInputHandler() : Base::InputServerBase
- AttachManager() : Game::FeatureUnit
- AttachMapper() : Resources::ResourceManager
- AttachmentManager() : GraphicsFeature::AttachmentManager
- AttachmentServer() : InternalGraphics::AttachmentServer
- AttachNode() : Models::Model
- AttachNodeInstance() : Models::ModelInstance
- AttachPort() : Messaging::Dispatcher
- AttachRenderEventHandler() : Graphics::Display
- AttachRequestHandler() : Http::HttpServerProxy, Http::HttpServer
- AttachTimeSource() : BaseGameFeature::TimeManager
- AttachVisibilityContainer() : Visibility::VisibilityChecker
- AttachVisibilitySystem() : Visibility::VisibilityChecker, Visibility::VisibilityQuery
- AttachVisibilitySystems() : Visibility::VisibilityChecker, InternalGraphics::InternalStage
- AttachVisibleLight() : Lighting::LightServerBase,
Lighting::LightPrePassServer
- AttachVisibleModelInstance() : Models::VisResolver
- AttachVisibleModelInstancePlayerCamera() : Models::VisResolver
- AutoEvade() : PhysicsFeature::ActorPhysicsProperty
- AutoManageManagedResource() : Resources::ResourceManager
- AxisAsString() : Base::GamePadBase
- b -

- r
- s
- t
- u
- v
- w
- x
- y
- z
- ~

b(): Math::plane
Back(): Util::Array< TYPE >, Util::List< TYPE >, Util::RingBuffer< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::Array< TYPE >, Util::List< TYPE >
barycentric(): Math::float4, Math::quaternion, Math::float4
BatchMessage(): Messaging::BatchMessage
bbox(): Math::bbox
Begin(): Http::HtmlPageWriter, Direct3D9::D3D9ShaderInstance, Util::Crc, Util::FixedArray< TYPE >, Util::Array< TYPE >, Util::List< TYPE >, Debug::DebugCounter, Util::Array< TYPE >, Direct3D9::D3D9ShaderInstance, Base::ShaderInstanceBase
begin_extend(): Math::bbox
BeginAddCategoryAttrs(): BaseGameFeature::CategoryManager
BeginAttach(): Direct3D9::D3D9ParticleRenderer, Base::ParticleRendererBase, Direct3D9::D3D9ParticleRenderer
BeginAttachVisibilityContainer(): Visibility::VisibilitySystemBase, Visibility::VisibilityChecker, Visibility::VisibilitySystemBase
BeginAttachVisibleLights(): Lighting::LightServerBase
BeginBatch(): Base::MultipleRenderTargetBase, Base::RenderTargetBase,
Base::RenderDeviceBase, Base::RenderTargetBase
- BeginBulkAdd() : Util::Dictionary< KEYTYPE, VALUETYPE >
- BeginCapture() : Input::InputHandler, Base::KeyboardBase, Base::MouseBase, Base::KeyboardBase, Base::MouseBase
  Input::InputHandler
- BeginDraw() : Characters::CharacterServer
- BeginEvents() : CoreAnimation::AnimClip
- BeginFrame() : Direct3D9::D3D9RenderDevice, Base::InputServerBase, Lighting::LightServerBase, Base::InputServerBase, Lighting::LightServerBase, Characters::CharacterServer, Lighting::LightServerBase, Direct3D9::D3D9RenderDevice, Base::RenderTargetBase
- BeginGather() : Characters::CharacterServer
- BeginGatherSkins() : Base::SkinnedMeshRendererBase
- BeginNode() : IO::XmlWriter, Http::SvgPageWriter
- BeginPaintGroup() : Http::SvgPageWriter
- BeginParseDataTags() : Models::ModelNode
- BeginPass() : Direct3D9::D3D9RenderTarget, Direct3D9::D3D9ShaderInstance, Base::RenderTargetBase, Base::MultipleRenderTargetBase, Base::RenderTargetBase, Base::MultipleRenderTargetBase, Base::RenderTargetBase, Direct3D9::D3D9ShaderInstance, Base::RenderTargetBase, Base::ShaderInstanceBase
- BeginResolve() : Models::VisResolver
- BeginSection() : Script::InfoLog
- BeginSetup() : Util::SparseTable< TYPE >
- BeginTextGroup() : Http::SvgPageWriter
- BeginTransformGroup() : Http::SvgPageWriter
- BinaryReader() : IO::BinaryReader
- BinarySearchIndex() : Util::FixedArray< TYPE >, Util::Array< TYPE >
- BinarySize() : OSX::OSXGuid, Win32::Win32Guid
- BinaryWriter() : IO::BinaryWriter
- Bind() : Base::ShaderVariableInstanceBase, Win360::Win360Socket, Messaging::DelegateTable, Win360::Win360Socket, Base::ShaderVariableInstanceBase
- BitField() : Util::BitField< NUMBITS >
- BitsPerPixel() : Resources::D3D9TextureStreamer
- Blob() : Util::Blob
- BlockingHandlerThread() : Messaging::BlockingHandlerThread
- Broadcast() : Net::StdTcpServer
- BuildSignature() : Base::VertexLayoutBase
- BuildStateMachineRacks() : Script::ScriptManager
- ButtonAsString() : Base::GamePadBase
- ButtonDoubleClicked() : Base::MouseBase
- ButtonDown() : Base::MouseBase, Base::GamePadBase, Base::MouseBase, Base::GamePadBase, Base::MouseBase
- ButtonPressed() : Base::MouseBase, Base::GamePadBase, Base::MouseBase
- ButtonUp() : Base::MouseBase, Base::GamePadBase, Base::MouseBase, Base::GamePadBase
- BXmLoaderUtil() : IO::BXmLoaderUtil
- BXmReader() : IO::BXmReader
- ByteOrder() : System::ByteOrder

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
- c -

- c() : **Math::plane**
- CalendarTimeBase() : **Base::CalendarTimeBase**
- CameraEntity() : **Graphics::CameraEntity**
- CameraProperty() : **GraphicsFeature::CameraProperty**
- CameraSettings() : **Shared::CameraSettings**
- CanBeMapped() : **IO::Stream**, **IO::ZipFileStream**, **IO::FileStream**, **IO::MemoryStream**
- Cancel() : **Messaging::AsyncPort**
- CancelCurrentDialog() : **Script::DialogManager**
- CancelMessage() : **FrameSync::FrameSyncHandlerThread**, **Messaging::HandlerThreadBase**, **Messaging::BlockingHandlerThread**, **Messaging::RunThroughHandlerThread**
- CanCreate() : **Base::RenderDeviceBase**, **Direct3D9::D3D9RenderDevice**
- CanRead() : **IO::ZipFileStream**, **IO::FileStream**,
IO::MemoryStream, IO::Stream
- CanSeek() : IO::Stream, IO::ZipFileStream, IO::FileStream, IO::MemoryStream
- CanWrite() : IO::Stream, IO::ZipFileStream, IO::FileStream, IO::MemoryStream
- Capacity() : Util::Array< TYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::Array< TYPE >, Util::RingBuffer< TYPE >
- cast() : Ptr< TYPE >
- Category() : BaseGameFeature::CategoryManager::Category, BaseGameFeature::CategoryManager::Entry
- CategoryManager() : BaseGameFeature::CategoryManager
- catmullrom() : Math::float4
- center() : Math::bbox
- centerX() : Math::rectangle< TYPE >
- centerY() : Math::rectangle< TYPE >
- ChangeFileExtension() : Util::String
- Character() : Characters::Character
- CharacterAnimationController() : Characters::CharacterAnimationController
- CharacterInstance() : Characters::CharacterInstance
- CharacterJoint() : Characters::CharacterJoint
- CharacterNode() : Characters::CharacterNode
- CharacterNodeInstance() : Characters::CharacterNodeInstance
- CharacterServer() : Characters::CharacterServer
- CharacterSkeleton() : Characters::CharacterSkeleton
- CharacterSkeletonInstance() : Characters::CharacterSkeletonInstance
- CharacterSkin() : Characters::CharacterSkin
- CharacterSkinLibrary() : Characters::CharacterSkinLibrary
- CharacterSkinList() : Characters::CharacterSkinList
- CharacterSkinNode() : Characters::CharacterSkinNode
- CharacterSkinNodeInstance() : Characters::CharacterSkinNodeInstance
- CharacterSkinSet() : Characters::CharacterSkinSet
- CharacterVariationSet() : Characters::CharacterVariationSet
- CharJointInfo() : Shared::CharJointInfo
- ChaseCameraProperty() : GraphicsFeature::ChaseCameraProperty
- CheckDebugRendering() : Game::GameServer
- CheckDone() : Jobs::TPJobPort, Jobs::SerialJobPort, Base::JobPortBase, Jobs::TPJobPort
- CheckFileExtension() : Util::String
- CheckId() : Messaging::Message
- CheckLodDistance() : Models::TransformNode
- CheckPendingResources() : Models::Model, Resources::ResourceManager
- CheckUpdateDone() : Characters::CharacterInstance
- CheckValidCharSet() : Util::String
- Child() : Util::SimpleTree< VALUETYPE >::Node
- ChooseResponse() : Script::DialogManager
- Circle() : Http::SvgPageWriter
- clamp() : Math::float4
- ClampKeyIndex() : CoreAnimation::AnimUtil
- ClassExists() : Core::Factory
- Cleanup() : BaseGameFeature::EntityManager, Base::ShaderInstanceBase, Direct3D9::D3D9ShaderInstance
- CleanupGameFeatures() : App::GameApplication
- CleanUpLoadingQueue() : Resources::ResourceScheduler
- CleanupStateHandlers() : App::GameApplication
- Clear() : Threading::SafeFlag, Threading::SafePriorityQueue< PRITYPE, TYPE >, Threading::SafeQueue< TYPE >, IO::URI, Util::Array< TYPE >, Util::BitField< NUMBITS >, Resources::ManagedResource, Util::FixedArray< TYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::FixedTable< TYPE >, Resources::ManagedResource, Util::HashTable< KEYTYPE, VALUETYPE >, Resources::ManagedResource, Util::Queue< TYPE >, Util::List< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::SparseTable< TYPE >, Util::Stack< TYPE >, Util::String, Util::StringAtom, Util::Variant, Characters::CharacterSkinSet, Util::List< TYPE >, Resources::ManagedResource, IO::MediaType, Util::Array< TYPE >
- ClearAssign() : IO::AssignRegistry
- ClearAttachments() : GraphicsFeature::AttachmentManager
- ClearAttachmentsOnEntity() : GraphicsFeature::AttachmentManager
- ClearBit() : Util::BitField< NUMBITS >
- ClearCapture() : Base::InputServerBase
- ClearEntities() : PhysicsFeature::TriggerProperty
- ClearEntity() : Game::Property
- ClearEnvEntity() : BaseGameFeature::EnvEntityManager
- ClearFeatureBits() : Base::ShaderServerBase
- ClearHandlers() : Messaging::HandlerThreadBase
- ClearKeyboardCapture() : Base::InputServerBase
- ClearLinks() : InternalGraphics::InternalGraphicsEntity
- ClearMouseCapture() : Base::InputServerBase
- ClearRenderStats() : Resources::ManagedResource
- ClearVisibilityLinks() : Visibility::VisibilityChecker
- Client() : FrameSync::FrameSyncSharedData
- ClientDiscard() : FrameSync::FrameSyncSharedData
- ClientSetup() : FrameSync::FrameSyncSharedData
- clip() : Math::frustum, Math::plane
- clipmask() : Math::frustum
- clipstatus() : Math::bbox, Math::frustum, Math::sphere
- Close() : InternalGraphics::AttachmentServer, InternalGraphics::InternalGraphicsServer, Models::ModelServer, Direct3D9::D3D9RenderDevice, Core::CoreServer, Lighting::SM30LightServer, Lighting::LightServerBase, IO::XmlReader, Lighting::SM30ShadowServer, Base::ShaderServerBase, Lighting::LightPrePassServer, Messaging::AsyncPort, IO::StreamReader, IO::StreamWriter, IO::StreamReader, IO::StreamWriter, IO::BXmlReader, IO::XmlWriter, IO::Console, Messaging::AsyncPort, IO::Stream, IO::StreamReader, Particles::ParticleServer, Lighting::LightPrePassServer, Resources::ResourceManager, IO::LogFileConsoleHandler, Lighting::SM30ShadowServer, Visibility::VisibilityChecker, Win360::Win360Socket, Visibility::VisibilityBoxSystem, Visibility::VisibilityClusterSystem, IO::ConsoleHandler, App::GameApplication, Visibility::VisibilitySystemBase, IO::BinaryWriter, IO::ZipFileEntry, IO::ConsoleHandler, IO::ExcelXmlReader, IO::ZipFileStream, IO::StreamWriter, Http::HtmlPageWriter, Game::GameServer, Messaging::AsyncPort, Http::SvgPageWriter, Animation::AnimEventServer, App::RenderApplication,
App::ViewerApplication, Http::HttpMessageHandler,
FSM::StateMachine, IO::StreamReader,
Win32::Win32InputServer, Http::HttpServer, IO::FileStream,
IO::IoInterfaceHandler, Models::VisResolver,
Messaging::Handler, IO::StreamWriter,
Win32::Win32InputServer, Base::InputServerBase,
IO::BinaryReader, Win360::Win360Socket,
Http::HttpServerProxy, Http::SvgPageWriter,
IO::StreamWriter, Base::ShapeRendererBase,
Base::DisplayDeviceBase, Base::RendererDeviceBase,
Debug::DebugHandler, Direct3D9::D3D9RenderDevice,
Win32::Win32DisplayDevice, Base::TextRendererBase,
Win32::Win32DisplayDevice, Base::TransformDeviceBase,
Base::VertexLayoutServerBase, Visibility::VisibilityQuadtree,
App::Application, Net::StdTcpServer,
Direct3D9::D3D9ShaderServer, Messaging::Handler,
Direct3D9::D3D9TextRenderer, Win32::Win32DisplayDevice,
Win360::D3D9ShapeRenderer, IO::ConsoleHandler,
Direct3D9::D3D9ShaderServer,
Direct3D9::D3D9TextRenderer, Graphics::GraphicsHandler,
IO::ConsoleHandler, BaseGameFeature::LoaderServer,
Win360::D3D9TransformDevice,
Base::VertexLayoutServerBase,
Win360::D3D9ShapeRenderer,
Win360::D3D9TransformDevice, Messaging::Handler,
IO::MemoryStream, Graphics::Display,
Messaging::AsyncPort, Graphics::GraphicsInterface,
Graphics::GraphicsServer, IO::ConsoleHandler,
Frame::FrameServer, Net::StdTcpServer,
Messaging::AsyncPort, App::ConsoleApplication

- CloseDInputMouse() : Win32::Win32InputServer
- CloseFile() : OSX::OSXFSWrapper, Win360::Win360FSWrapper
- CloseProgressIndicator() : BaseGameFeature::LoaderServer
- closestpoint() : Math::line
- CloseTask() : Script::Task
- CloseWindow() : Win32::Win32DisplayDevice
- Column() : Util::QuadTree< TYPE >::Node
- CommandLineArgs() : Util::CommandLineArgs
- Commit() : Direct3D9::D3D9ShaderInstance, Base::ShaderInstanceBase, Direct3D9::D3D9ShaderInstance
- CommitChangesToDatabase() : BaseGameFeature::CategoryManager
- CompareExchange() : OSX::OSXInterlocked, Win360::Win360Interlocked
- Compute() : Lighting::PSSMUtil, Util::Crc
- ComputeAbsMousePos() : Win32::Win32DisplayDevice
- ComputeAdjustedWindowRect() : Win32::Win32DisplayDevice
- ComputeAlignedBlockSize() : OSX::OSXMemoryPool, Win360::Win360MemoryPool
- ComputeBlendWeight() : Animation::AnimJob
- ComputeDecodedSize() : Util::RunLengthCodec
- ComputeFileCrc() : IO::IoServer
- ComputeKeySlicePointerAndSize() : CoreAnimation::AnimResource
- ComputeMinMaxAvgCounts() : Debug::DebugPageHandler
- ComputeMinMaxAvgTimes() : Debug::DebugPageHandler
- ComputeMouseWorldRay() : BaseGameFeature::EnvQueryManager
- ComputeNormMousePos() : Win32::Win32DisplayDevice
- ComputeWorldMouseRay() : RenderUtil::MouseRayUtil
- Concatenate() : Util::String
- Condition() : Conditions::Condition
- ConfigureAnimDrivenMotionTracking() : InternalGraphics::InternalModelEntity, Graphics::ModelEntity
- ConfigureAnimEventTracking() : InternalGraphics::InternalModelEntity, Graphics::ModelEntity
- ConfigureCharJointTracking() : InternalGraphics::InternalModelEntity,
Graphics::ModelEntity
  - Confirm() : IO::Console, IO::ConsoleHandler, Win32::Win32ConsoleHandler, IO::ConsoleHandler, OSX::OSXConsoleHandler, IO::ConsoleHandler
  - conjugate() : Math::quaternion
  - Connect() : Net::MessageClientConnection, Win360::Win360Socket, Net::StdTcpClient, Net::StdTcpClientConnection, Win360::Win360Socket, Net::MessageClient, Net::StdTcpClientConnection
  - Console() : IO::Console
  - ConsoleApplication() : App::ConsoleApplication
  - ConsoleHandler() : IO::ConsoleHandler
  - ConsolePageHandler() : Debug::ConsolePageHandler
  - ConstructContent() : Script::ScriptManager
  - ConsumeNewModelNodeInstanceIndex() : Models::ModelServer
  - Contains() : Util::Queue<TYPE>, Util::HashTable<KEYTYPE, VALUETYPE>, Util::Dictionary<KEYTYPE, VALUETYPE>, Util::Queue<TYPE>, Util::Stack<TYPE>
  - contains() : Math::bbox
  - ContainsCharFromSet() : Util::String
  - ContainsDescription() : Script::InfoLogFilter
  - ContainsInformation() : Script::InfoLogFilter
  - ContainsLogLevel() : Script::InfoLogFilter
  - ContainsSource() : Script::InfoLogFilter
  - Content() : Util::HashTable<KEYTYPE, VALUETYPE>
  - Continue() : BaseGameFeature::TimeSource
  - ContinueAll() : BaseGameFeature::TimeManager
  - ContinueFollow() : PhysicsFeature::ActorPhysicsProperty
  - ContinueGoto() : PhysicsFeature::ActorPhysicsProperty
  - Convert() : System::ByteOrder
  - ConvertBackslashes() : Util::String
  - ConvertInPlace() : System::ByteOrder
  - ConvertPath() : OSX::OSXFSWrapper
  - ConvertToArchiveURI() : IO::ArchiveBase, IO::ZipArchive
  - ConvertToPathInArchive() : IO::ZipArchive, IO::ArchiveBase
  - CopyFile() : IO::IoServer
  - CopyToBuffer() : Util::String
  - CorePageHandler() : Debug::CorePageHandler
CoreServer() : Core::CoreServer
corner_point() : Math::bbox
CpuTypeAsString() : Base::SystemInfoBase
Crc() : Util::Crc
Create() : Core::Factory, Core::Rtti, Core::Factory
CreateActionFromString() : Actions::Action
CreateActionsFromString() : Actions::Action
CreateAndSetupGraphicsEntities() :

GraphicsFeature::SegmentedGfxUtil
CreateAsSingleton() : Debug::DebugTimer
CreateCollisionShape() : PhysicsFeature::TriggerProperty
CreateConditionFromString() : Conditions::Condition
CreateConditionsFromString() : Conditions::Condition
CreateDialog() : Script::DialogManager
CreateDialogTake() : Script::Dialog
CreateDirectory() : OSX::OSXFSWrapper, Win360::Win360FSWrapper, IO::IoServer
CreateDummyInstance() :

BaseGameFeature::CategoryManager
CreateEntityByAttrs() : BaseGameFeature::FactoryManager
CreateEntityByCategory() :

BaseGameFeature::FactoryManager
CreateEntityByClassName() :

BaseGameFeature::FactoryManager
CreateEntityByEntity() : BaseGameFeature::FactoryManager
CreateEntityByEntityAsCategory() :

BaseGameFeature::FactoryManager
CreateEntityByGuid() : BaseGameFeature::FactoryManager
CreateEntityByKeyAttr() : BaseGameFeature::FactoryManager
CreateEntityByTemplate() :

BaseGameFeature::FactoryManager
CreateEntityByTemplateAsCategory() :

BaseGameFeature::FactoryManager
CreateEntityCommand() : Commands::CreateEntityCommand
CreateEnvEntity() : BaseGameFeature::EnvEntityManager
CreateEvaluationJob() : Animation::AnimJob
CreateInstance() : Models::Model, Base::ShaderVariableBase
CreateInstanceFromAttrs() :

BaseGameFeature::CategoryManager
- CreateInstanceFromInstance() : BaseGameFeature::CategoryManager
- CreateInstanceFromInstanceAsCategory() : BaseGameFeature::CategoryManager
- CreateInstanceFromTemplate() : BaseGameFeature::CategoryManager
- CreateInstanceFromTemplateAsCategory() : BaseGameFeature::CategoryManager
- CreateManagedResource() : Resources::ResourceManager
- CreateNodeInstance() : Models::TransformNode, Models::StateNode, Characters::CharacterNode, Characters::CharacterSkinNode, Models::ShapeNode, Models::ModelNode, Particles::ParticleSystemNode
- CreateNodeInstanceHierarchy() : Models::ModelNode
- CreatePartialInstance() : Models::Model
- CreatePhysicsEntity() : PhysicsFeature::ActorPhysicsProperty
- CreateProperty() : BaseGameFeature::FactoryManager
- CreateSampleAndMixJob() : CoreAnimation::AnimUtil
- CreateSampleJob() : CoreAnimation::AnimUtil
- CreateShaderInstance() : Base::ShaderServerBase, Base::ShaderBase, Base::ShaderServerBase, Base::ShaderBase
- CreateShaderVariableInstance() : Models::StateNodeInstance
- CreateSharedVertexLayout() : Base::VertexLayoutServerBase
- CreateStream() : IO::IoServer
- CreateUserProfile() : BaseGameFeature::LoaderServer
- CreateVisibilityJob() : Visibility::VisibilityBoxSystem, Visibility::VisibilityClusterSystem, Visibility::VisibilitySystemBase, Visibility::VisibilityQuadtree
- cross3() : Math::float4
- CurveByIndex() : CoreAnimation::AnimClip
- d -

- d() : Math::plane
- D3D9DisplayDevice() : Direct3D9::D3D9DisplayDevice
- D3D9IndexBuffer() : Win360::D3D9IndexBuffer
- D3D9ParticleRenderer() : Direct3D9::D3D9ParticleRenderer
- D3D9ParticleSystemInstance() :
  Direct3D9::D3D9ParticleSystemInstance
- D3D9RenderDevice() : Direct3D9::D3D9RenderDevice
- D3D9RenderTarget() : Direct3D9::D3D9RenderTarget
- D3D9Shader() : Direct3D9::D3D9Shader
- D3D9ShaderInstance() : Direct3D9::D3D9ShaderInstance
- D3D9ShaderServer() : Direct3D9::D3D9ShaderServer
- D3D9ShaderVariable() : Direct3D9::D3D9ShaderVariable
- D3D9ShaderVariation() : Direct3D9::D3D9ShaderVariation
- D3D9ShapeRenderer() : Win360::D3D9ShapeRenderer
- D3D9StreamMeshLoader() : Win360::D3D9StreamMeshLoader
- D3D9TextRenderer() : Direct3D9::D3D9TextRenderer
- D3D9Texture() : Direct3D9::D3D9Texture
- D3D9TransformDevice() : Win360::D3D9TransformDevice
- D3D9VertexBuffer() : Win360::D3D9VertexBuffer
- D3D9VertexLayout() : Win360::D3D9VertexLayout
- DeactivateDialog() : Script::Dialog
- DeactivateEntity() : BaseGameFeature::EntityManager
- DebugCounter() : Debug::DebugCounter
- DebugGraphicsHandler() : Debug::DebugGraphicsHandler
- DebugHandler() : Debug::DebugHandler
- DebugInterface() : Debug::DebugInterface
- DebugMessage() : Net::DebugMessage
- DebugOut() : IO::LogFileConsoleHandler, OSX::OSXConsoleHandler, Win32::Win32ConsoleHandler, OSX::SysFunc, IO::Console, Win32::SysFunc, IO::ConsoleHandler, IO::HistoryConsoleHandler
- DebugPacket() : Net::DebugPacket
- DebugPageHandler() : Debug::DebugPageHandler
- DebugServer() : Debug::DebugServer
- DebugShapeRenderer() : Debug::DebugShapeRenderer
- DebugTextRenderer() : Debug::DebugTextRenderer
- DebugTimer() : Debug::DebugTimer
- Decode() : Messaging::Message, Util::RunLengthCodec, Messaging::Message
- DecodePackets() : Net::DebugPacket
- DecodeStream() : Net::TcpMessageCodec
- decompose() : Math::matrix44
- Decr() : Debug::DebugCounter
- DecrClientCount() : Resources::ManagedResource
- Decrement() : OSX::OSXInterlocked, Win360::Win360Interlocked
- DecrUseCount() : Resources::Resource
- DeferredHandled() : Messaging::Message
- DelayedJob() : BaseGameFeature::EntityManager::DelayedJob
- Delegate() : Util::Delegate< ARGTYPE >
- Delete() : System::Win32Registry
- DeleteDirectory() : OSX::OSXFSWrapper, Win360::Win360FSWrapper, IO::IoServer
- DeleteEntity() : BaseGameFeature::EntityManager
- DeleteEntityImmediate() : BaseGameFeature::EntityManager
- DeleteEnvEntity() : BaseGameFeature::EnvEntityManager
- DeleteFile() : OSX::OSXFSWrapper, IO::IoServer, Win360::Win360FSWrapper
- DeleteInstance() : BaseGameFeature::CategoryManager
- DeleteLevel() : BaseGameFeature::CategoryManager
- DeleteProfile() : BaseGameFeature::UserProfile
- DeleteShapes():
- DebugPacket() : Net::DebugPacket
- DelayedJob() : BaseGameFeature::CategoryManager
- DeferredHandled() : Messaging::Message
- Decompose() : Math::matrix44
PhysicsFeature::EnvironmentCollideProperty
- DeleteShapesByThreadId() : Base::ShapeRendererBase
- DeleteTextElementsByThreadId() : Base::TextRendererBase
- Dequeue() : Threading::SafePriorityQueue< PRITYPE, TYPE >, Threading::SafeQueue< TYPE >, Util::Queue< TYPE >
- DequeueAll() : Threading::SafeQueue< TYPE >
- DequeueMessages() : Net::TcpMessageCodec
- Destroy() : Core::Factory
- DestroyCollisionShape() : PhysicsFeature::TriggerProperty
- DestroySingleton() : Debug::DebugTimer
- DetachEntity() : InternalGraphics::AttachmentServer
- determinant() : Math::matrix44
- diagonal_size() : Math::bbox
- Dialog() : Script::Dialog
- DialogDesc() : Script::DialogDesc
- DialogExistsById() : Script::DialogManager
- DialogManager() : Script::DialogManager
- DialogTake() : Script::DialogTake
- DialogTakeExistsById() : Script::DialogManager
- Dictionary() : Util::Dictionary< KEYTYPE, VALUETYPE >
- Difference() : Util::Array< TYPE >
- DirectoryExists() : OSX::OSXFSWrapper, Win360::Win360FSWrapper, IO::IoServer
- DisablePhysics() : PhysicsFeature::ActorPhysicsProperty, PhysicsFeature::PhysicsProperty
- Discard() : Direct3D9::D3D9RenderTarget, Debug::DebugTimer, Jobs::TPJobSystem, Base::JobBase, Graphics::GraphicsEntity, Debug::DebugCounter, Graphics::GraphicsEntity, IO::SchemeRegistry, FrameSync::FrameSyncTimer, IO::BXmlLoaderUtil, Graphics::GraphicsEntity, Win360::D3D9VertexLayout, Models::StateNodeInstance, Base::SkinnedMeshRendererBase, Frame::FrameBatch, Frame::FramePass, Base::SkinnedMeshRendererBase, Frame::FramePassBase, Frame::FramePostEffect, IO::ArchiveBase, Frame::FrameShader, Graphics::CameraEntity, IO::ZipArchive, IO::ZipFileSystem, Graphics::GraphicsEntity, Util::StringBuffer, Animation::AnimSequencer, Base::JobPortBase,
DiscardAllStages() : Graphics::GraphicsServer, InternalGraphics::InternalGraphicsServer
DiscardHierarchy() : Models::ModelNodeInstance
DiscardInstance() : Models::Model
- DiscardManagedModel() : Models::ModelServer
- DiscardManagedResource() : Resources::ResourceManager
- DiscardShaderInstance() : Base::ShaderBase
- Disconnect() : Net::MessageClient, Net::StdTcpClient
- Dispatch() : Messaging::StaticMessageHandler
- Dispatcher() : Messaging::Dispatcher
- Display() : Graphics::Display
- DisplayDevice() : CoreGraphics::DisplayDevice
- DisplayDeviceBase() : Base::DisplayDeviceBase
- DisplayEvent() : CoreGraphics::DisplayEvent
- DisplayEventHandler() : CoreGraphics::DisplayEventHandler
- DisplayMode() : CoreGraphics::DisplayMode, Graphics::DisplaySettings
- DisplayPageHandler() : Debug::DisplayPageHandler
- DisplaySettings() : Graphics::DisplaySettings
- distance() : Math::line
- DoCollideCheck() : GraphicsFeature::ChaseCameraProperty
- DoResourceLOD() : Resources::ResourceScheduler, Resources::TexturePoolMapperScheduler
- DoStateTransition() : App::GameApplication
- dot() : Math::plane, Math::quaternion
- dot3() : Math::float4
- downcast() : Ptr< TYPE >
- DoWork() : Messaging::Handler, Win360::Win360Thread, Messaging::RunThroughHandlerThread, Win360::Win360Thread, FrameSync::FrameSyncHandlerThread, Debug::DebugGraphicsHandler, OSX::OSXThread, Http::HttpMessageHandler, Win360::Win360Thread, Interface::InterfaceHandlerBase, Graphics::GraphicsHandler, Debug::DebugHandler, Interface::InterfaceHandlerBase, Jobs::TPWorkerThread, Messaging::BlockingHandlerThread
- Draw() : Direct3D9::D3D9RenderDevice, Base::RenderDeviceBase, Http::SvgLineChartWriter, Direct3D9::D3D9RenderDevice,
RenderUtil::DrawFullScreenQuad
- DrawBox() : Debug::DebugShapeRenderer
- DrawCylinder() : Debug::DebugShapeRenderer
- DrawFullScreenQuad() : RenderUtil::DrawFullScreenQuad
- DrawGPUSkinnedMesh() : Characters::CharacterServer, Base::SkinnedMeshRendererBase
- DrawGPUTextureSkinnedMesh() : Base::SkinnedMeshRendererBase, Characters::CharacterServer
- DrawIndexedInstanced() : Direct3D9::D3D9RenderDevice, Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
- DrawIndexedPrimitives() : Debug::DebugShapeRenderer
- DrawPrimitives() : Debug::DebugShapeRenderer
- DrawShapes() : Win360::D3D9ShapeRenderer, Base::ShapeRendererBase
- DrawSoftwareSkinnedMesh() : Base::SkinnedMeshRendererBase, Characters::CharacterServer
- DrawSphere() : Debug::DebugShapeRenderer
- DrawText() : Debug::DebugTextRenderer
- DrawText3D() : Debug::DebugTextRenderer
- DrawTextElements() : Direct3D9::D3D9TextRenderer, Base::TextRendererBase
- DrawTorus() : Debug::DebugShapeRenderer
- DumpLeaks() : Core::RefCountedList
- DumpRefCountingLeaks() : Core::RefCounted
- DuplicateLevel() : BaseGameFeature::CategoryManager

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
 a  
 b  
 c  
 d  
 e  
 f  
 g  
 h  
 i  
 j  
 k  
 l  
 m  
 n  
 o  
 p  
 q
- e -

- Element() : **Http::HtmlPageWriter**
- Ellipse() : **Http::SvgPageWriter**
- EmitAnimEvents() : **Animation::AnimJob**, **Animation::AnimSequencer**
- EmitterAttrs() : **Particles::EmitterAttrs**
- EmitterMesh() : **Particles::EmitterMesh**
- EmitWakeupSignal() : **Win360::Win360Thread**, **Messaging::BlockingHandlerThread**, **OSX::OSXThread**, **Win360::Win360Thread**, **Jobs::TPWorkerThread**, **Win360::Win360Thread**
- EnablePhysics() : **PhysicsFeature::ActorPhysicsProperty**, **PhysicsFeature::PhysicsProperty**
- Encode() : **Messaging::Message**, **Util::RunLengthCodec**, **Messaging::Message**
- EncodeStream() : **Net::DebugPacket**
- EncodeToMessage() : **Net::TcpMessageCodec**
- End() : **Util::Crc**, **Util::Array< TYPE >**, **Util::FixedArray< TYPE >**, **Util::Array< TYPE >**
- end() : **Math::line**
- End() : **Util::List< TYPE >**, **Http::HtmlPageWriter**, **Debug::DebugCounter**, **Base::ShaderInstanceBase**, **Util::List< TYPE >**, **Direct3D9::D3D9ShaderInstance**
- end_extend() : **Math::bbox**
- EndAddCategoryAttrs() : **BaseGameFeature::CategoryManager**
EndAttach() : Direct3D9::D3D9ParticleRenderer, Base::ParticleRendererBase
EndAttachVisibilityContainer() : Visibility::VisibilitySystemBase, Visibility::VisibilityChecker, Visibility::VisibilityBoxSystem
EndAttachVisibleLights() : Lighting::LightServerBase
EndBatch() : Base::MultipleRenderTargetBase, Base::RenderDeviceBase, Base::RenderTargetBase, Base::MultipleRenderTargetBase, Base::RenderDeviceBase, Base::RenderTargetBase
EndBulkAdd() : Util::Dictionary<KEYTYPE, VALUETYPE>
EndCapture() : Input::InputHandler, Base::KeyboardBase, Base::MouseBase, Base::KeyboardBase, Input::InputHandler
EndDraw() : Characters::CharacterServer
EndEvents() : CoreAnimation::AnimClip
EndFrame() : Base::InputServerBase, Lighting::LightServerBase, Lighting::LightPrePassServer, Base::InputServerBase, Characters::CharacterServer, Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice, Base::InputServerBase, Direct3D9::D3D9RenderDevice, Lighting::LightPrePassServer, Lighting::LightServerBase
EndGather() : Characters::CharacterServer
EndGatherSkins() : Base::SkinnedMeshRendererBase
EndGroup() : Http::SvgPageWriter
EndNode() : Http::SvgPageWriter, IO::XmlWriter
EndParseDataTags() : Models::ModelNode
EndPass() : Direct3D9::D3D9ShaderInstance, Base::RenderDeviceBase, Base::MultipleRenderTargetBase, Base::RenderTargetBase, Base::ShaderInstanceBase, Direct3D9::D3D9RenderDevice, Direct3D9::D3D9RenderTarget, Direct3D9::D3D9ShaderInstance, Direct3D9::D3D9RenderTarget, Base::MultipleRenderTargetBase
EndResolve() : Models::VisResolver
EndSection() : Script::InfoLog
EndSetup() : Util::SparseTable<TYPE>
Enqueue() : Util::Queue<TYPE>, Threading::SafeQueue<TYPE>
- EnqueueAnimJob() : Animation::AnimSequencer, Characters::CharacterAnimationController
- EnqueueArray() : Threading::SafeQueue< TYPE >
- Enter() : OSX::OSXCriticalSection, Win360::Win360CriticalSection
- EnterLockStepMode() : FrameSync::FrameSyncHandlerThread, Graphics::GraphicsInterface
- Entity() : Game::Entity
- EntityIsInActiveLayer() : BaseGameFeature::EntityLoaderBase
- EntityLoaderBase() : BaseGameFeature::EntityLoaderBase
- EntityManager() : BaseGameFeature::EntityManager
- Entry() : BaseGameFeature::CategoryManager::Entry, Resources::ResourceDictionary::Entry
- EnumProfiles() : BaseGameFeature::UserProfile
- EnvelopeCurve() : Particles::EnvelopeCurve
- EnvelopeSampleBuffer() : Particles::EnvelopeSampleBuffer
- EnvEntityExists() : BaseGameFeature::EnvEntityManager
- EnvEntityManager() : BaseGameFeature::EnvEntityManager
- EnvironmentCollideProperty() : PhysicsFeature::EnvironmentCollideProperty
- EnvQueryManager() : BaseGameFeature::EnvQueryManager
- Eof() : IO::StreamReader, IO::FileStream, Win360::Win360FSWrapper, IO::StreamReader, IO::Stream, IO::StreamReader, OSX::OSXFSWrapper, IO::ZipFileStream, IO::MemoryStream
- equal3_all() : Math::float4
- equal3_any() : Math::float4
- equal4_all() : Math::float4
- equal4_any() : Math::float4
- Erase() : Util::Array< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::Array< TYPE >
- EraseAtIndex() : Util::Dictionary< KEYTYPE, VALUETYPE >
- EraseIndex() : Util::Array< TYPE >
- EraseIndexSwap() : Util::Array< TYPE >
- EraseMatchingElements() : Threading::SafePriorityQueue< PRITYPE, TYPE >, Threading::SafeQueue< TYPE >
- EraseSwap() : Util::Array< TYPE >
- ExtractRange() : Util::String
- ExtractToEnd() : Util::String
- ExtractToLastSlash() : Util::String
- Extrapolator() : Math::Extrapolator< TYPE >
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Enumerator
- Related Functions

- _
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
FactoryManager() : BaseGameFeature::FactoryManager
FailTask() : Script::Task
FeatureMaskToString() : Base::ShaderServerBase
FeatureStringToMask() : Base::ShaderServerBase
FeatureUnit() : Game::FeatureUnit
FileExists() : Win360::Win360FSWrapper, IO::IoServer, OSX::OSXFSWrapper
FileStream() : IO::FileStream
FileTimeToLocalTime() : Base::CalendarTimeBase, Win360::Win360CalendarTime
FileTimeToSystemTime() : Base::CalendarTimeBase, Win360::Win360CalendarTime
Fill() : Util::Array<TYPE>, Util::FixedArray<TYPE>
FillActionAndConditionLists() : Script::ScriptManager
Filter() : Script::InfoLogFilter
FilterByDescription() : Script::InfoLogFilter
FilterByInformation() : Script::InfoLogFilter
FilterByLogLevel() : Script::InfoLogFilter
FilterBySource() : Script::InfoLogFilter
FilterEntities() : PhysicsFeature::TriggerProperty
Find() : Util::List<TYPE>, Util::SimpleTree<VALUETYPE>::Node, Util::List<TYPE>, Util::Array<TYPE>, Util::StringAtomTableBase, Util::Array<TYPE>, Util::FixedArray<TYPE>
- FindArchiveWithDir() : IO::ZipFileSystem
- FindArchiveWithFile() : IO::ZipFileSystem
- FindAttrIndex() : IO::BXmlLoaderUtil
- FindBluePrint() : BaseGameFeature::FactoryManager
- FindCharIndex() : Util::String
- FindChildNodeIndex() : IO::BXmlLoaderUtil
- FindColumnIndex() : IO::ExcelXmlReader
- FindComponent() : Base::VertexLayoutBase
- FindContainmentNode() : Util::QuadTree< TYPE >::Node
- FindDirEntry() : IO::ZipDirEntry
- FindFileEntry() : IO::ZipDirEntry
- FindGuidIndex() : Script::ScriptManager
- FindIndex() : Util::Array< TYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::FixedArray< TYPE >
- FindInstances() : BaseGameFeature::CategoryManager
- FindNodeIndex() : IO::BXmlLoaderUtil
- FindProperty() : Game::Entity
- FindSiblingNodeIndex() : IO::BXmlLoaderUtil
- FindStateByName() : FSM::StateMachine
- FindStateHandlerByName() : App::GameApplication
- FindStringIndex() : Util::String
- FindSubTaskByGuid() : Script::Task
- FindSubTaskById() : Script::Task
- FindTemplate() : BaseGameFeature::CategoryManager
- FindTemplateByAttr() : BaseGameFeature::CategoryManager
- FirstCharToUpper() : Util::String
- FixedArray() : Util::FixedArray< TYPE >
- FixedTable() : Util::FixedTable< TYPE >
- FixFadeTimes() : Animation::AnimJob
- float2() : Math::float2
- float4() : Math::float4
- Flush() : IO::FileStream, Win360::Win360FSWrapper, IO::Stream, OSX::OSXFSWrapper, IO::Stream
- FlushBatchedMessages() : Graphics::GraphicsInterface
- FocusManager() : BaseGameFeature::FocusManager
- Format() : Base::CalendarTimeBase, Util::String
- FormatArgList() : Util::String
- FourCC() : Util::FourCC
- Fragment() : IO::URI
- FrameBatch() : Frame::FrameBatch
- FramePass() : Frame::FramePass
- FramePassBase() : Frame::FramePassBase
- FramePostEffect() : Frame::FramePostEffect
- FrameServer() : Frame::FrameServer
- FrameShader() : Frame::FrameShader
- FrameSyncHandlerThread() :
  FrameSync::FrameSyncHandlerThread
- FrameSyncSharedData() : FrameSync::FrameSyncSharedData
- FrameSyncTimer() : FrameSync::FrameSyncTimer
- Free() : OSX::OSXHeap, Win360::Win360Heap, Win360::Win360MemoryPool, Memory::PoolArrayAllocator, OSX::OSXMemoryPool
- FreeInstanceMemory() : Core::Rtti
- FreeJointTextureRow() : Base::SkinnedMeshRendererBase
- From() : Util::String
- FromBinary() : Win32::Win32Guid, OSX::OSXGuid
- FromBool() : Util::String
- FromFloat() : Util::String
- FromFloat2() : Util::String
- FromFloat4() : Util::String
- FromFunction() : Util::Delegate< ARGTYPE >
- FromInt() : Util::String
- FromList() : Actions::SequenceAction
- FromMatrix44() : Util::String
- FromMediaType() : CoreGraphics::ImageFileFormat
- FromMethod() : Util::Delegate< ARGTYPE >
- FromName() : Models::ModelNodeType
- FromString() : Conditions::Operator< TYPE >, Animation::AnimJobEnqueueMode, Http::HttpStatusCode, CoreAnimation::CurveType, Util::FourCC, CoreGraphics::Adapter, OSX::OSXGuid, CoreGraphics::ImageFileFormat, Http::HttpMethod, Win32::Win32Guid, CoreGraphics::PrimitiveTopology, CoreGraphics::AntiAliasQuality, Lighting::LightType, Input::MouseButton, Frame::SortingMode, CoreGraphics::PixelFormat, Input::Key, CoreGraphics::BatchType, Frame::LightingMode, CoreAnimation::SampleType, CoreAnimation::InfinityType,
CoreGraphics::IndexType

- Front() : Util::List< TYPE >, Util::Array< TYPE >, Util::List< TYPE >, Util::Array< TYPE >, Util::RingBuffer< TYPE >, Util::SimpleTree< VALUETYPE >::Node
- frustum() : Math::frustum
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q
- g -

- GameApplication() : App::GameApplication
- GameContentServer() : IO::GameContentServer
- GameContentServerBase() : Base::GameContentServerBase
- GamePadBase() : Base::GamePadBase
- GameServer() : Game::GameServer
- GameStateHandler() : BaseGameFeature::GameStateHandler
- GameThreadWaitForFrameSync() : Graphics::GraphicsInterface
- GatherSkinMesh() : Characters::CharacterServer
- GatherVisibleCharacter() : Characters::CharacterServer
- ge() : Math::float2
- gen() : Math::noise
- Generate() : OSX::OSXGuid, Win32::Win32Guid
- GenerateMipLevels() : Base::RenderTargetBase, Direct3D9::D3D9RenderTarget
- get() : WeakPtr< TYPE >, OSX::OSXThreadLocalPtr< TYPE >
- Get() : IO::BXmlReader, IO::XmlReader, Util::String
- get() : Ptr< TYPE >
- Get() : Models::VisResolveContainer< TYPE >
- get_cartesian() : Math::polar
- get_clipplanes() : Math::bbox
- get_position() : Math::matrix44
- get_unsafe() : WeakPtr< TYPE >, Ptr< TYPE >
- get_xaxis() : Math::matrix44
- get_yaxis() : `Math::matrix44`
- get_zaxis() : `Math::matrix44`
- GetAbsMousePos() : `CoreGraphics::DisplayEvent`, `Input::InputEvent`
- GetAbsoluteEndTime() : `Animation::AnimJob`
- GetAbsoluteStartTime() : `Animation::AnimJob`
- GetAbsoluteStopTime() : `Animation::AnimJob`
- GetAcceptedMessages() : `Messaging::Port`
- GetAccess() : `Win360::D3D9StreamMeshLoader`, `Base::ResourceBase`, `CoreGraphics::MemoryMeshLoader`, `Win360::D3D9StreamMeshLoader`, `Base::ResourceBase`
- GetAccessMode() : `IO::Stream`
- GetAccessPattern() : `IO::Stream`
- GetAction() : `Script::ActionReader`
- GetActionBock() : `Script::DialogTake`
- GetActionList() : `Actions::SequenceAction`, `Actions::ActionList`, `Script::DialogTake`
- GetActionRef() : `Script::DialogTake`
- GetActionString() : `Script::ActionReader`
- GetActiveShaderInstance() : `Base::ShaderServerBase`
- GetActiveVariation() : `Base::ShaderInstanceBase`, `Characters::CharacterVariationSet`, `Base::ShaderInstanceBase`
- GetAdapter() : `Graphics::DisplaySettings`, `Base::DisplayDeviceBase`
- GetAdapterInfo() : `Direct3D9::D3D9DisplayDevice`, `Graphics::Display`, `Base::DisplayDeviceBase`, `Direct3D9::D3D9DisplayDevice`, `Base::DisplayDeviceBase`
- GetAddress() : `Win360::Win360Socket`, `Net::StdTcpServer`, `Win360::Win360Socket`, `Net::StdTcpServer`
- GetAlignedBlockSize() : `OSX::OSXMemoryPool`, `Win360::Win360MemoryPool`
- GetAllAnimJobs() : `Animation::AnimSequencer`
- GetAllAssigns() : `IO::AssignRegistry`
- GetAllRegisteredUriSchemes() : `IO::SchemeRegistry`
- GetAllShaderInstances() : `Base::ShaderBase`
- GetAllShaders() : `Base::ShaderServerBase`
- GetAlpha() : `CoreGraphics::MousePointer`
- GetAmbientLightColor() : Graphics::GlobalLightEntity, Lighting::InternalGlobalLightEntity
- GetAnimationResourceId() : Characters::CharacterNode
- GetAnimDrivenMotionJointName() : Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- GetAnimDrivenMotionVector() : Graphics::ModelEntity, Characters::CharacterAnimationController, InternalGraphics::InternalModelEntity
- GetAnimEvent() : Animation::AnimEventInfo
- GetAnimEvents() : Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- GetAnimEventsByCategory() : BaseGameFeature::AnimEventManager
- GetAnimJobName() : Animation::AnimEventInfo
- GetAnimJobsByName() : Animation::AnimSequencer
- GetAnimJobsByTrackIndex() : Animation::AnimSequencer
- GetAntiAliasQuality() : Base::RenderTargetBase, Graphics::DisplaySettings, Base::DisplayDeviceBase, Base::RenderTargetBase, Base::DisplayDeviceBase
- GetAppDataDirectory() : Win360::Win360FSWrapper, OSX::OSXFSWrapper
- GetAppID() : App::Application
- GetAppName() : Interface::InterfaceHandlerBase, Interface::InterfaceBase, Interface::InterfaceHandlerBase, Interface::InterfaceBase, Core::CoreServer, Interface::InterfaceBase, Interface::InterfaceHandlerBase, Interface::InterfaceBase, Interface::InterfaceHandlerBase, Interface::InterfaceBase, Interface::InterfaceHandlerBase
- GetAppTitle() : App::Application
- GetAppVersion() : App::Application
- GetAspect() : Shared::CameraSettings
- GetAspectRatio() : CoreGraphics::DisplayMode
- GetAssign() : IO::AssignRegistry
- GetAttr() : Game::Entity
- GetAttrName() : IO::BXmLLoaderUtil
- GetAttrs() : IO::BXmLReader, IO::XmlReader
- GetAttrTable() : Game::Entity
- GetAttrTableRowIndex() : Game::Entity
- `GetAttrValue()`: `IO::BXmlLoaderUtil`
- `GetAvailableDisplayModes()`: `Direct3D9::D3D9DisplayDevice`, `Graphics::Display`, `Base::DisplayDeviceBase`, `Direct3D9::D3D9DisplayDevice`, `Base::DisplayDeviceBase`
- `GetAxisValue()`: `Base::GamePadBase`
- `GetBackColor()`: `Graphics::GlobalLightEntity`, `Lighting::InternalGlobalLightEntity`
- `GetBackLightColor()`: `Graphics::GlobalLightEntity`, `Lighting::InternalGlobalLightEntity`
- `GetBackLightOffset()`: `Graphics::GlobalLightEntity`, `Lighting::InternalGlobalLightEntity`
- `GetBaseIndex()`: `CoreGraphics::PrimitiveGroup`
- `GetBaseTime()`: `Animation::AnimJob`
- `GetBaseVertex()`: `CoreGraphics::PrimitiveGroup`
- `GetBatchByIndex()`: `Frame::FramePassBase`
- `GetBinDirectory()`: `Win360::Win360FSWrapper`, `OSX::OSXFSWrapper`
- `GetBlendWeight()`: `Animation::AnimJob`
- `GetBlob()`: `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`, `Util::Variant`
- `GetBlobArray()`: `Util::Variant`
- `GetBlocking()`: `Win360::Win360Socket`
- `GetBlockSize()`: `OSX::OSXMemoryPool`, `Win360::Win360MemoryPool`
- `GetBool()`: `BaseGameFeature::UserProfile`, `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`, `Script::ActionReader`, `IO::BXmlReader`, `IO::XmlReader`, `Util::CommandLineArgs`, `Util::Variant`, `Particles::EmitterAttrs`
- `GetBoolArray()`: `Util::Variant`
- `GetBoolAtIndex()`: `Util::CommandLineArgs`
- `GetBoolFlag()`: `Util::CommandLineArgs`
- `GetBoundingBox()`: `CoreGraphics::PrimitiveGroup`, `Util::QuadTree< TYPE >::Node`, `Models::ModelNode`, `Models::Model`, `Models::ModelNode`, `Visibility::ObserverContext`, `Visibility::VisibilityContext`
- `GetBroadcast()`: `Win360::Win360Socket`
- `GetBuffer()`: `Util::RingBuffer< TYPE >`
- `GetBufferSize()`: `Jobs::JobDataDesc`, `Jobs::JobUniformDesc`
- `GetByteSize()`: `CoreAnimation::AnimKeyBuffer`
- `GetCameraEntity()`: `Graphics::View`,
**InternalGraphics::InternalView**, **Lighting::PSSMUtil**

- GetCameraFocusEntity() : **BaseGameFeature::FocusManager**
- GetCameraSettings() : **Graphics::CameraEntity**, **InternalGraphics::InternalCameraEntity**
- GetCameraTransform() : **RenderUtil::MayaCameraUtil**
- GetCastShadows() : **Graphics::AbstractLightEntity**, **Lighting::InternalAbstractLightEntity**
- GetCastShadowsThisFrame() : **Lighting::InternalAbstractLightEntity**
- GetCategory() : **Game::Entity**, **Characters::CharacterSkin**, **CoreAnimation::AnimEvent**
- GetCategoryByIndex() : **BaseGameFeature::CategoryManager**
- GetCategoryByName() : **BaseGameFeature::CategoryManager**
- GetCategoryName() : **Animation::AnimEventHandlerBase**
- GetChar() : **CoreGraphics::DisplayEvent**, **Input::InputEvent**
- GetCharacter() : **InternalGraphics::InternalModelEntity**, **Characters::CharacterNode**
- GetCharacterInstance() : **InternalGraphics::InternalModelEntity**, **Characters::CharacterNodeInstance**
- GetCharInput() : **Base::KeyboardBase**
- GetCharJointInfos() : **InternalGraphics::InternalModelEntity**
- GetChildAt() : **Util::QuadTree<TYPE>::Node**
- GetChildren() : **Models::ModelNode**, **Models::ModelNodeInstance**, **Models::ModelNode**, **Models::ModelNodeInstance**, **Models::ModelNode**, **Models::ModelNodeInstance**, **Models::ModelNode**
- GetChildTakes() : **Script::DialogTake**
- GetClass() : **Script::ActionReader**
- GetClassFourCC() : **Core::RefCounted**
- GetClassName() : **Core::RefCounted**
- GetClassRtti() : **Core::Factory**
- GetClearColor() : **Base::RenderTargetBase**, **Frame::FramePassBase**, **Base::RenderTargetBase**
- GetClearDepth() : **Base::RenderTargetBase**, **Frame::FramePassBase**, **Base::MultipleRenderTargetBase**, **Base::RenderTargetBase**, **Base::MultipleRenderTargetBase**
- GetClearFlags() : **Base::RenderTargetBase**, **Frame::FramePassBase**, **Base::MultipleRenderTargetBase**, **Base::RenderTargetBase**
Frame::FramePassBase, Base::RenderTargetBase
- Frame::FramePassBase, Base::RenderTargetBase, Base::RenderTargetBase, Base::RenderTargetBase, Base::RenderTargetBase
- Frame::FramePassBase, Base::RenderTargetBase, Base::RenderTargetBase, Base::RenderTargetBase
- Frame::FramePassBase, Base::RenderTargetBase, Base::RenderTargetBase, Base::RenderTargetBase

- GetClearStencil() : Base::RenderTargetBase, Frame::FramePassBase, Base::RenderTargetBase, Base::RenderTargetBase
- GetClearStencil() : Base::RenderTargetBase, Frame::FramePassBase, Base::RenderTargetBase, Base::RenderTargetBase
- GetClearStencil() : Base::RenderTargetBase, Frame::FramePassBase, Base::RenderTargetBase, Base::RenderTargetBase
- GetClearStencil() : Base::RenderTargetBase, Frame::FramePassBase, Base::RenderTargetBase, Base::RenderTargetBase

- GetClientAddress() : Net::StdTcpClientConnection
- GetClientAddress() : Net::StdTcpClientConnection
- GetClientAddress() : Net::StdTcpClientConnection
- GetClientAddress() : Net::StdTcpClientConnection

- GetClientConnectionClass() : Net::StdTcpServer
- GetClientConnectionClass() : Net::StdTcpServer
- GetClientConnectionClass() : Net::StdTcpServer
- GetClientConnectionClass() : Net::StdTcpServer

- GetClientCount() : Resources::ManagedResource
- GetClientCount() : Resources::ManagedResource
- GetClientCount() : Resources::ManagedResource
- GetClientCount() : Resources::ManagedResource

- GetClipByIndex() : CoreAnimation::AnimResource
- GetClipByIndex() : CoreAnimation::AnimResource
- GetClipByIndex() : CoreAnimation::AnimResource
- GetClipByIndex() : CoreAnimation::AnimResource

- GetClipByName() : CoreAnimation::AnimResource
- GetClipByName() : CoreAnimation::AnimResource
- GetClipByName() : CoreAnimation::AnimResource
- GetClipByName() : CoreAnimation::AnimResource

- GetClipDuration() : CoreAnimation::AnimClip
- GetClipDuration() : CoreAnimation::AnimClip
- GetClipDuration() : CoreAnimation::AnimClip
- GetClipDuration() : CoreAnimation::AnimClip

- GetClipIndexByName() : CoreAnimation::AnimResource
- GetClipIndexByName() : CoreAnimation::AnimResource
- GetClipIndexByName() : CoreAnimation::AnimResource
- GetClipIndexByName() : CoreAnimation::AnimResource

- GetClipName() : Animation::PlayClipJob
- GetClipName() : Animation::PlayClipJob
- GetClipName() : Animation::PlayClipJob
- GetClipName() : Animation::PlayClipJob

- GetCloseConditions() : Graphics::GraphicsEntity
- GetCloseConditions() : Graphics::GraphicsEntity
- GetCloseConditions() : Graphics::GraphicsEntity
- GetCloseConditions() : Graphics::GraphicsEntity

- GetCloseText() : Script::Task
- GetCloseText() : Script::Task
- GetCloseText() : Script::Task
- GetCloseText() : Script::Task

- GetCmdLineArgs() : Game::FeatureUnit, App::Application, Game::FeatureUnit, App::Application, Game::FeatureUnit
- GetCmdLineArgs() : Game::FeatureUnit, App::Application, Game::FeatureUnit, App::Application

- GetCmdName() : Util::CommandLineArgs
- GetCmdName() : Util::CommandLineArgs

- GetCode() : Jobs::TPJobCommand
- GetCode() : Jobs::TPJobCommand

- GetColor() : CoreGraphics::RenderShape, CoreGraphics::TextElement, Graphics::AbstractLightEntity, Lighting::InternalAbstractLightEntity
- GetColor() : CoreGraphics::RenderShape, CoreGraphics::TextElement, Graphics::AbstractLightEntity, Lighting::InternalAbstractLightEntity

- GetColorBufferFormat() : Base::RenderTargetBase
- GetColorBufferFormat() : Base::RenderTargetBase

- GetColumnIndexByName() : Util::SparseTable< TYPE >
- GetColumnIndexByName() : Util::SparseTable< TYPE >

- GetCompanyName() : App::Application, Interface::InterfaceHandlerBase, Interface::InterfaceBase, App::Application, Interface::InterfaceHandlerBase, App::Application, Interface::InterfaceHandlerBase, Interface::InterfaceBase, Interface::InterfaceHandlerBase, Interface::InterfaceBase, Interface::InterfaceHandlerBase, Interface::InterfaceBase, Interface::InterfaceHandlerBase, Interface::InterfaceBase, Interface::InterfaceHandlerBase

- GetComponentAt() : Base::VertexLayoutBase
- GetComponentAt() : Base::VertexLayoutBase

- GetCondition() : Script::ActionReader, Actions::IfThenElseAction, Conditions::Not, Script::DialogTake
- GetCondition() : Script::ActionReader, Actions::IfThenElseAction, Conditions::Not, Script::DialogTake

- GetConditions() : Conditions::And, Conditions::Or
- GetConditions() : Conditions::And, Conditions::Or

- GetContent() : IO::XmlReader
- GetContent() : IO::XmlReader

- GetContentLength() : Http::HttpResponseReader
- GetContentLength() : Http::HttpResponseReader

- GetContentType() : Http::HttpResponseReader
- GetContentType() : Http::HttpResponseReader
- GetCoreId(): OSX::OSXThread, Win360::Win360Thread
- GetCornerIndexBuffer(): Direct3D9::D3D9ParticleRenderer
- GetCornerVertexBuffer(): Direct3D9::D3D9ParticleRenderer
- GetCount(): Net::DebugPacket
- GetCPUResolveTexture(): Base::RenderTargetBase
- GetCpuType(): Base::SystemInfoBase
- GetCurParticleVertexIndex(): Direct3D9::D3D9ParticleRenderer
- GetCurrentAdapterDisplayMode(): Direct3D9::D3D9DisplayDevice, Base::DisplayDeviceBase, Graphics::Display, Direct3D9::D3D9DisplayDevice, Base::DisplayDeviceBase
- GetCurrentGlobalLightEntity(): InternalGraphics::InternalGraphicsServer
- GetCurrentNodeIndex(): IO::BXmlLoaderUtil
- GetCurrentNodeLineNumber(): IO::XmlReader
- GetCurrentNodeName(): IO::BXmlLoaderUtil, IO::XmlReader, IO::BXmlReader
- GetCurrentNodePath(): IO::BXmlReader, IO::XmlReader
- GetCurrentState(): FSM::StateMachine, App::GameApplication
- GetCurrentStateHandler(): App::GameApplication
- GetCurrentTake(): Script::Dialog
- GetCurVertexPtr(): Direct3D9::D3D9ParticleRenderer
- GetCurveType(): CoreAnimation::AnimCurve
- GetD3D9BaseTexture(): Direct3D9::D3D9Texture
- GetD3D9CubeTexture(): Direct3D9::D3D9Texture
- GetD3D9Effect(): Direct3D9::D3D9ShaderVariation, Direct3D9::D3D9Shader, Direct3D9::D3D9ShaderInstance, Direct3D9::D3D9ShaderVariation, Direct3D9::D3D9Shader, Direct3D9::D3D9ShaderInstance
- GetD3D9EffectPool(): Direct3D9::D3D9ShaderServer
- GetD3D9IndexBuffer(): Win360::D3D9IndexBuffer
- GetD3D9Technique(): Direct3D9::D3D9ShaderVariation
- GetD3D9Texture(): Direct3D9::D3D9Texture
- GetD3D9VertexBuffer(): Win360::D3D9VertexBuffer
- GetD3D9VertexDeclaration(): Win360::D3D9VertexLayout
- GetD3D9VolumeTexture(): Direct3D9::D3D9Texture
- GetDatabasePath(): BaseGameFeature::UserProfile
- GetDay() : Base::CalendarTimeBase
- GetDebugColor() : FSM::StateMachine
- GetDebugCounterByNamed() : Debug::DebugServer
- GetDebugCounters() : Debug::DebugServer
- GetDebugInfo() : Util::GlobalStringAtomTable
- GetDebugTextEnabled() : BaseGameFeature::LoaderServer
- GetDebugTimerByNamed() : Debug::DebugServer
- GetDebugTimers() : Debug::DebugServer
- GetDebugTxt() : Actions::Action , FSM::StateMachine , Actions::Action
- GetDefaultGamePad() : Base::InputServerBase
- GetDefaultKeyboard() : Base::InputServerBase
- GetDefaultMouse() : Base::InputServerBase
- GetDefaultRenderTarget() : Base::RenderTargetBase
- GetDefaultSamplesArray() : Characters::CharacterSkeleton
- GetDependencies() : InternalGraphics::InternalView
- GetDepth() : Base::TextureBase
- GetDepthResolveTexture() : Base::RenderTargetBase
- GetDesc() : Http::HttpRequestHandler
- GetDescription() : Script::InfoLog , CoreGraphics::AdapterInfo
- GetDeviceId() : CoreGraphics::AdapterInfo
- GetDeviceIndex() : Input::InputEvent
- GetDeviceName() : CoreGraphics::AdapterInfo
- GetDialog() : Script::DialogManager
- GetDialogId() : Script::DialogManager
- GetDialogLockState() : Script::DialogManager
- GetDialogTakeState() : Script::DialogManager
- GetDialogText() : Script::DialogManager
- GetDialogTexts() : Script::Dialog
- GetDirect3D() : Direct3D9::D3D9RenderDevice
- GetDirect3DDevice() : Direct3D9::D3D9RenderDevice
- GetDirEntries() : IO::ZipDirEntry
- GetDisplayMode() : Base::DisplayDeviceBase
- GetDriverName() : CoreGraphics::AdapterInfo
- GetDriverVersionHighPart() : CoreGraphics::AdapterInfo
- GetDriverVersionLowPart() : CoreGraphics::AdapterInfo
- GetDuration() : Animation::AnimJob
- GetElement() : **Util::QuadTree< TYPE >::Node**, **IO::ExcelXmlReader**
- GetElements() : **Util::SparseTable< TYPE >**
- GetElseBlock() : **Actions::IfThenElseAction**
- GetEmitterAttrs() : **Particles::ParticleSystem**, **Particles::ParticleSystemNode**
- GetEmitterMesh() : **Particles::ParticleSystem**
- GetEmitterMeshResourceId() : **Particles::ParticleSystemNode**
- GetEmitterPoint() : **Particles::EmitterMesh**
- GetEmote() : **Script::DialogTake**
- GetEnqueueMode() : **Animation::AnimJob**
- GetEntities() : **BaseGameFeature::EntityManager**, **InternalGraphics::InternalGraphicsServer**, **InternalGraphics::InternalStage**
- GetEntitiesByAttr() : **BaseGameFeature::EntityManager**
- GetEntitiesByAttrs() : **BaseGameFeature::EntityManager**
- GetEntitiesByType() : **InternalGraphics::InternalStage**
- GetEntitiesInActivityBubble() : **BaseGameFeature::EntityManager**
- GetEntitiesInBox() : **BaseGameFeature::EnvQueryManager**
- GetEntitiesInSphere() : **BaseGameFeature::EnvQueryManager**
- GetEntity() : **Game::Property**, **Script::ActionReader**, **Actions::Action**, **Conditions::Condition**, **FSM::StateMachine**, **Game::Property**
- GetEntityByAttr() : **BaseGameFeature::EntityManager**
- GetEntityByAttrs() : **BaseGameFeature::EntityManager**
- GetEntityById() : **InternalGraphics::InternalGraphicsServer**
- GetEntityByUniqueId() : **BaseGameFeature::EntityManager**
- GetEntityId() : **Animation::AnimEventInfo**
- GetEntityUnderMouse() : **BaseGameFeature::EnvQueryManager**
- GetEntry() : **Resources::ResourceDictionary**
- GetEnvelope() : **Particles::EmitterAttrs**
- GetEnvelopeSampleBuffer() : **Particles::ParticleSystem**
- GetErrorCode() : **Win360::Win360Socket**
- GetErrorString() : **Win360::Win360Socket**
- GetEventByIndex(): `CoreAnimation::AnimClip`
- GetEventByIndex(): `CoreAnimation::AnimClip`
- GetEventCode(): `CoreGraphics::DisplayEvent`, `CoreGraphics::RenderEvent`
- GetEventsInRange(): `CoreAnimation::AnimClip`
- GetExclusiveTag(): `Animation::AnimJob`
- GetFactor(): `BaseGameFeature::TimeSource`
- GetFadeInTime(): `Animation::AnimJob`
- GetFadeOutTime(): `Animation::AnimJob`
- GetFailConditions(): `Script::Task`
- GetFailedText(): `Script::Task`
- GetFarHeight(): `Shared::CameraSettings`
- GetFarWidth(): `Shared::CameraSettings`
- GetFeatureBits(): `Base::ShaderServerBase`
- GetFeatureMask(): `Base::ShaderVariationBase`
- GetFileEntries(): `IO::ZipDirEntry`
- GetFileExtension(): `Util::String`
- GetFileName(): `StateObjectFeature::StateProperty`
- GetFileSize(): `Win360::Win360FSWrapper`, `IO::ZipFileEntry`, `OSX::OSXFSWrapper`
- GetFileWriteTime(): `Win360::Win360FSWrapper`, `IO::IoServer`, `OSX::OSXFSWrapper`
- GetFirstKeyIndex(): `CoreAnimation::AnimCurve`
- GetFirstSlice(): `Jobs::TPJobCommand`
- GetFittingPoolIndex(): `Resources::PoolResourceMapper`
- GetFloat(): `BaseGameFeature::UserProfile`, `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`, `Script::ActionReader`, `IO::XmlReader`, `Util::CommandLineArgs`, `IO::BXmlReader`, `Util::Variant`, `Particles::EmitterAttrs`
- GetFloat2(): `IO::XmlReader`, `IO::BXmlReader`
- GetFloat4(): `BaseGameFeature::UserProfile`, `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`, `Script::ActionReader`, `IO::XmlReader`, `Util::CommandLineArgs`, `IO::BXmlReader`, `Util::Variant`
- GetFloat4Array(): `Util::Variant`
- GetFloat4AtIndex(): `Util::CommandLineArgs`
- GetFloatArray(): `Util::Variant`
- GetFloatAtIndex(): `Util::CommandLineArgs`
- `GetFocalLength()`: `Base::TransformDeviceBase`, `Shared::CameraSettings`
- `GetFocusEntity()`: `BaseGameFeature::FocusManager`
- `GetFormat()`: `Base::StreamTextureSaverBase`
- `GetFourCC()`: `Core::Rtti`
- `GetFov()`: `Shared::CameraSettings`
- `GetFragmentJointPalette()`: `Characters::CharacterSkinNode`
- `GetFragmentPrimGroupIndex()`: `Characters::CharacterSkinNode`
- `GetFrameActions()`: `FSM::State`
- `GetFrameCount()`: `FrameSync::FrameSyncHandlerThread`, `FrameSync::FrameSyncTimer`, `Graphics::GraphicsServer`
- `GetFrameId()`: `BaseGameFeature::TimeSource`
- `GetFramePassBaseByIndex()`: `Frame::FrameShader`
- `GetFramePassBaseByName()`: `Frame::FrameShader`
- `GetFrameShader()`: `InternalGraphics::InternalView`
- `GetFrameShaderName()`: `Graphics::View`
- `GetFrameTicks()`: `FrameSync::FrameSyncTimer`
- `GetFrameTime()`: `FrameSync::FrameSyncTimer`, `App::RenderApplication`, `BaseGameFeature::TimeSource`, `App::RenderApplication`, `BaseGameFeature::TimeSource`
- `GetFrequency()`: `GraphicsFeature::LightFlickerUtil`
- `GetFromByteOrder()`: `System::ByteOrder`
- `GetFuncDesc()`: `Base::JobBase`
- `GetFunctionPointer()`: `Jobs::SerialJobFuncDesc`, `Jobs::TPJobFuncDesc`
- `GetGfxEntity()`: `Visibility::VisibilityContext`
- `GetGlobalLightShadowBufferTexture()`: `Lighting::SM30ShadowServer`
- `GetGlobalMatrix()`: `Shared::CharJointInfo`
- `GetGlobalStringBuffer()`: `Util::GlobalStringAtomTable`
- `GetGraphicsEntities()`: `BaseGameFeature::EnvEntityManager`, `Visibility::VisibilityContainer`
- `GetGraphicsResource()`: `GraphicsFeature::GraphicsProperty`
- `GetGroup()`: `Script::Dialog`, `Script::DialogDesc`
- `GetGuid()`: `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`, `Script::Dialog`, `Script::DialogTake`,
Script::DialogDesc, Script::Task, Util::Variant, CoreGraphics::AdapterInfo
- GetGuidArray() : Util::Variant
- GetHandlerAtIndex() : Messaging::Port
- GetHandlers() : IO::Console
- GetHandlerThread() : Messaging::AsyncPort
- GetHeight() : Base::RenderTargetBase, Base::TextureBase, CoreGraphics::DisplayMode, Base::RenderTargetBase
- GetHighFrequencyVibrator() : Base::GamePadBase
- GetHistory() : Debug::DebugTimer, IO::HistoryConsoleHandler, Debug::DebugCounter
- GetHistorySize() : IO::HistoryConsoleHandler
- GetHomeDirectory() : Win360::Win360FSWrapper, OSX::OSXFSWrapper
- GetHostAddr() : Win360::Win360IpAddress
- GetHostAndLocalPath() : IO::URI
- GetHostName() : Win360::Win360IpAddress
- GetHotspot() : CoreGraphics::MousePointer
- GetHour() : Base::CalendarTimeBase
- GetHttpMethod() : Http::HttpRequestReader
- GetHwnd() : Win32::Win32DisplayDevice
- GetIconName() : Graphics::DisplaySettings, Base::DisplayDeviceBase
- GetId() : Script::Dialog, Script::DialogDesc, Script::DialogTake, Script::Task, InternalGraphics::InternalGraphicsEntity
- GetIndexBuffer() : Base::RenderDeviceBase, Base::MeshBase, Base::RenderDeviceBase, Base::MeshBase
- GetIndexData() : CoreGraphics::RenderShape
- GetIndexType() : Base::IndexBufferBase, CoreGraphics::RenderShape, Base::IndexBufferBase
- GetInfo() : Script::InfoLog
- GetInput() : Win32::Win32ConsoleHandler, IO::ConsoleHandler, IO::Console, IO::ConsoleHandler, OSX::OSXConsoleHandler, IO::ConsoleHandler
- GetInputDesc() : Base::JobBase
- GetInputFocusEntity() : BaseGameFeature::FocusManager
- GetInstanceDataset() : BaseGameFeature::CategoryManager::Category
- GetInstanceEntity() : BaseGameFeature::CategoryManager
- GetInstances() : Models::Model
- GetInstancesByAttr() : BaseGameFeature::CategoryManager
- GetInstancesByAttrs() : BaseGameFeature::CategoryManager
- GetInstanceSize() : Core::Rtti
- GetInstanceTable() : BaseGameFeature::CategoryManager
- GetInstanceTableName() :
  BaseGameFeature::CategoryManager::Category
- GetInt() : BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity, Script::ActionReader, IO::XmlReader, Util::CommandLineArgs, IO::BXmlReader, Util::Variant, Particles::EmitterAttrs
- GetIntArray() : Util::Variant
- GetIntAtIndex() : Util::CommandLineArgs
- GetIntensityAmplitude() : GraphicsFeature::LightFlickerUtil
- GetInvLightProjTransform() :
  Lighting::InternalAbstractLightEntity
- GetInvModelTransform() : Base::TransformDeviceBase
- GetInvModelViewTransform() : Base::TransformDeviceBase
- GetInvPoseMatrix() : Characters::CharacterJoint
- GetInvPoseMatrixArray() : Characters::CharacterSkeleton
- GetInvProjTransform() : Base::TransformDeviceBase, Shared::CameraSettings
- GetInvTransform() :
  Lighting::InternalAbstractLightEntity
- GetJob() : Jobs::TPJobSlice
- GetJobPort() : Characters::CharacterInstance
- GetJoint() : Characters::CharacterSkeleton
- GetJointIndexByName() : Characters::CharacterSkeleton
- GetJointMatrix() : Characters::CharacterSkeletonInstance
- GetJointName() : Shared::CharJointInfo
- GetJointTextureRowIndex() : Characters::CharacterInstance
- GetKeepAlive() : Win360::Win360Socket
- GetKey() : CoreGraphics::DisplayEvent, Input::InputEvent
- GetKeyboardCaptureHandler() : Base::InputServerBase
- GetKeyBuffer() : CoreAnimation::AnimResource
- GetKeyBufferPointer() : CoreAnimation::AnimKeyBuffer
- GetKeyDuration() : CoreAnimation::AnimClip
- GetKeySliceByteSize() : CoreAnimation::AnimClip
- GetKeySliceFirstKeyIndex() : CoreAnimation::AnimClip
- GetKeyStride() : CoreAnimation::AnimClip
- GetLastFrameId() : Resources::ManagedResource
- GetLevelName() : BaseGameFeature::GameStateHandler
- GetLightDir() : Lighting::PSSMUtil
- GetLightDirection() : Lighting::InternalGlobalLightEntity
- GetLightEntity() : GraphicsFeature::LightFlickerUtil
- GetLightingMode() : Frame::FrameBatch
- GetLightType() : Graphics::AbstractLightEntity, Lighting::InternalAbstractLightEntity
- GetLinks() : InternalGraphics::InternalGraphicsEntity
- GetLoadedResource() : Resources::ManagedResource
- GetLoader() : Resources::Resource
- GetLocalBoundingBox() :
  InternalGraphics::InternalGraphicsEntity, Graphics::GraphicsEntity,
  InternalGraphics::InternalGraphicsEntity, Graphics::GraphicsEntity,
  InternalGraphics::InternalGraphicsEntity
- GetLocalLightShadowBufferTexture() : Lighting::SM30ShadowServer
- GetLocalMatrices() :
  PhysicsFeature::EnvironmentCollideProperty
- GetLocalMatrix() : Shared::CharJointInfo
- GetLocalTime() : Base::CalendarTimeBase, Win360::Win360CalendarTime
- GetLocalTransform() : Models::TransformNodeInstance
- GetLockedToViewer() : Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode,
  Models::TransformNodeInstance
- GetLogLevel() : Script::InfoLog
- GetLoopCount() : Animation::PlayClipJob
- GetLowFrequencyVibrator() : Base::GamePadBase
- GetMagic() : Net::DebugPacket
- GetMainRenderTarget() : Frame::FrameShader
- GetManagedAnimResource() : Characters::CharacterNode
- GetManagedMesh() : Models::ShapeNode
- GetManagedResource() : Resources::LoadingResource
- GetMapperByResourceType() : Resources::ResourceManager
- getmatrix() : Math::transform44
- GetMatrix44() : BaseGameFeature::GlobalAttrsManager, Game::Entity, Util::CommandLineArgs, IO::XmlReader, IO::BXmlReader, Util::Variant
- GetMatrix44Array() : Util::Variant
- GetMatrix44AtIndex() : Util::CommandLineArgs
- GetMaxDistance() : Models::TransformNode
- GetMaxMsgSize() : Win360::Win360Socket
- GetMaxNumLocalPlayers() : Base::InputServerBase
- GetMaxNumParticles() : Particles::ParticleSystem
- GetMaxNumPlayers() : Base::GamePadBase
- GetMaxProgressValue() : BaseGameFeature::LoaderServer
- GetMaxScreenSpaceSize() : Resources::ManagedResource
- GetMaxTriggerDistance() : BaseGameFeature::EntityManager
- GetMaxValue() : Particles::EnvelopeCurve
- GetMediaType() : IO::Stream
- GetMemoryPool() : Memory::PoolArrayAllocator
- GetMemorySize() : Base::RenderTargetBase
- GetMesh() : Resources::ManagedMesh
- GetMeshResourceId() : Models::ShapeNode
- GetMessages() : Messaging::BatchMessage
- GetMethod() : Http::HttpRequest
- GetMilliSecond() : Base::CalendarTimeBase
- GetMinDistance() : Models::TransformNode
- GetMinute() : Base::CalendarTimeBase
- GetMipLevel() : Base::StreamTextureSaverBase
- GetModel() : Models::ModelNode, Models::ModelInstance, Models::ModelNode, Models::ManagedModel, Models::ModelNode
- GetModelEntity() : Models::ModelInstance
- GetModelInstance() : Models::ModelNodeInstance, InternalGraphics::InternalModelEntity, Models::ModelNodeInstance
- GetModelNode() : Models::ModelNodeInstance, Characters::CharacterSkin, Models::ModelNodeInstance
- GetModelNodeInstanceIndex() : Models::ModelNodeInstance
- GetModelResourceMapper() : Models::ModelServer
- GetModelResourceState() : 
  InternalGraphics::InternalModelEntity
- GetModelTransform() : Models::TransformNodeInstance, 
  Base::TransformDeviceBase, CoreGraphics::RenderShape, 
  Models::TransformNodeInstance
- GetModelViewProjTransform() : Base::TransformDeviceBase
- GetModelViewTransform() : Base::TransformDeviceBase
- GetMonth() : Base::CalendarTimeBase
- GetMouseButton() : CoreGraphics::DisplayEvent, 
  Input::InputEvent
- GetMouseCaptureHandler() : Base::InputServerBase
- GetMouseMovement() : Win32::Win32InputServer
- GetMousePos3d() : BaseGameFeature::EnvQueryManager
- GetMovement() : Base::MouseBase, Win32::Win32Mouse
- GetMRTIndex() : Base::RenderTargetBase
- GetMultipleRenderTarget() : Frame::FramePassBase, 
  InternalGraphics::InternalView
- GetMultipleRenderTargetByIndex() : Frame::FrameShader
- GetMultipleRenderTargetByName() : Frame::FrameShader
- GetMyThreadId() : OSX::OSXThread, Win360::Win360Thread
- GetMyThreadName() : Win360::Win360Thread
- GetName() : BaseGameFeature::UserProfile, 
  Base::ShaderVariableBase, IO::Assign, 
  Base::ShaderVariableBase, Debug::DebugTimer, 
  BaseGameFeature::CategoryManager::Category, 
  Base::ShaderVariationBase, OSX::OSXHeap, 
  Win360::Win360Thread, Base::ShaderVariationBase, 
  Win360::Win360Thread, Http::HttpRequestHandler, 
  OSX::OSXThread, FSM::State, FSM::StateMachine, 
  Http::HttpRequestHandler, Win360::Win360Thread, 
  Http::HttpRequestHandler, Models::ModelNodeInstance, 
  Http::HttpRequestHandler, Core::Rtti, 
  Http::HttpRequestHandler, Frame::FramePassBase, 
  Animation::AnimJob, Models::ModelNodeInstance, 
  Win360::Win360Thread, Models::ModelNode, 
  Frame::FrameShader, IO::ZipDirEntry, IO::ZipFileEntry, 
  Characters::CharacterSkin, Http::HttpRequestHandler, 
  Models::ModelNode, Graphics::View, Models::ModelNode, 
  Win360::Win360Thread, Animation::AnimJob,
Models::ModelNode, Debug::DebugCounter, Http::HttpRequestHandler, Models::ModelNode, Characters::CharacterJoint, InternalGraphics::InternalStage, Models::ModelNodeInstance, InternalGraphics::InternalView, Base::ShaderVariableBase, Models::ModelNodeInstance, Win360::Win360Thread, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Characters::CharacterSkinList, Win360::Win360Thread, Models::ModelNode, Models::ModelNodeInstance, CoreAnimation::AnimClip, CoreAnimation::AnimEvent, Win360::Win360Heap, Models::ModelNode, Frame::FramePassBase
- GetNearHeight() : Shared::CameraSettings
- GetNearWidth() : Shared::CameraSettings
- GetNewSlot() : Resources::PoolLoadingResource
- GetNextThreadIndex() : Jobs::TPJobThreadPool
- GetNodeFilter() : Frame::FrameBatch
- GetNodeInstances() : Models::ModelInstance
- GetNoDelay() : Win360::Win360Socket
- GetNodeName() : IO::BXmlLoaderUtil
- GetNodes() : Models::Model
- GetNormMousePos() : Input::InputEvent, CoreGraphics::DisplayEvent
- GetNumArgs() : Util::CommandLineArgs
- GetNumAttrs() : IO::BXmlLoaderUtil
- GetNumBatches() : Frame::FramePassBase
- GetNumBlocks() : Win360::Win360MemoryPool, OSX::OSXMemoryPool
- GetNumBuffers() : Jobs::JobUniformDesc, Jobs::JobDesc
- GetNumCategories() : BaseGameFeature::CategoryManager
- GetNumChildTakes() : Script::DialogTake
- GetNumChunks() : Util::StringBuffer
- GetNumClips() : CoreAnimation::AnimResource
- GetNumColumns() : Util::SparseTable< TYPE >, IO::ExcelXmlReader
- GetNumComponents() : Base::VertexLayoutBase
- GetNumCpuCores() : Base::SystemInfoBase
- GetNumCurves() : CoreAnimation::AnimClip
- GetNumEvents(): CoreAnimation::AnimClip
- GetNumFragments(): Characters::CharacterSkinNode
- GetNumFramePassBases(): Frame::FrameShader
- GetNumHandlers(): Messaging::Port
- GetNumIndices(): Base::IndexBufferBase, CoreGraphics::PrimitiveGroup, Base::IndexBufferBase
- GetNumInstances(): BaseGameFeature::CategoryManager
- GetNumJoints(): Characters::CharacterSkeletonInstance, Characters::CharacterSkeleton
- GetNumKeys(): CoreAnimation::AnimClip, CoreAnimation::AnimKeyBuffer
- GetNumMipLevels(): Base::TextureBase
- GetNumMultipleRenderTargetTargets(): Frame::FrameShader
- GetNumPasses(): Base::ShaderVariationBase
- GetNumPendingResources(): Resources::SimpleResourceMapper, Resources::ResourceMapper
- GetNumPools(): Resources::PoolResourceMapper
- GetNumPrimitiveGroups(): Base::MeshBase
- GetNumPrimitives(): CoreGraphics::RenderShape, CoreGraphics::PrimitiveGroup
- GetNumRenderTargets(): Frame::FrameShader
- GetNumRendertargets(): Base::MultipleRenderTargetBase
- GetNumRows(): Util::SparseTable< TYPE >, IO::ExcelXmlReader, Resources::D3D9TextureStreamer
- GetNumSamples(): Particles::EnvelopeSampleBuffer, CoreAnimation::AnimSampleBuffer
- GetNumSharedVariables(): Base::ShaderServerBase
- GetNumSkinLists(): Characters::CharacterSkinLibrary
- GetNumSkins(): Characters::CharacterSkinSet, Characters::CharacterSkinLibrary
- GetNumSlices(): Jobs::TPJobCommand
- GetNumStates(): App::GameApplication
- GetNumSubTasks(): Script::Task
- GetNumTables(): IO::ExcelXmlReader
- GetNumTextures(): Frame::FrameShader
- GetNumTimeSources(): BaseGameFeature::TimeManager
- GetNumTransitions(): FSM::State
- GetNumUniqueElements(): Util::SparseTable< TYPE >
- GetNumVariables() : Base::ShaderInstanceBase, Frame::FramePassBase, Frame::FrameBatch, Frame::FramePassBase, Base::ShaderInstanceBase
- GetNumVariations() : Base::ShaderInstanceBase
- GetNumVertices() : Base::VertexBufferBase, CoreGraphics::PrimitiveGroup, CoreGraphics::RenderShape
- GetObject() : Util::Variant
- GetObjectRef() : Graphics::GraphicsEntity
- GetObserver() : Visibility::VisibilityQuery
- GetObserverBitMask() : Visibility::VisibilityBoxSystem, Visibility::VisibilityClusterSystem, Visibility::VisibilityQuadtree, Visibility::VisibilitySystemBase
- GetObserverEntity() : Visibility::ObserverContext
- GetObserverType() : Visibility::VisibilityQuery
- getoffset() : Math::transform44
- GetOffsetMatrix() : Models::TransformNodeInstance
- GetOldSlot() : Resources::PoolLoadingResource
- GetOpenConditions() : Script::Task
- GetOpenText() : Script::Task
- GetOptBool() : IO::BXmlReader, IO::XmlReader
- GetOptFloat() : IO::XmlReader, IO::BXmlReader
- GetOptFloat2() : IO::XmlReader, IO::BXmlReader
- GetOptFloat4() : IO::XmlReader, IO::BXmlReader
- GetOptInt() : IO::XmlReader, IO::BXmlReader
- GetOptMatrix44() : IO::XmlReader, IO::BXmlReader
- GetOptString() : IO::XmlReader, IO::BXmlReader
- GetOrientation() : CoreGraphics::MousePointer
- GetOriginalShader() : Base::ShaderInstanceBase
- GetOutputDesc() : Base::JobBase
- GetPageSize() : Base::SystemInfoBase
- GetParent() : Models::ModelNodeInstance, Core::Rtti, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode
- GetParentGuid() : Script::Task
- GetParentJoint() : Characters::CharacterJoint
- GetParentJointIndex() : Characters::CharacterJoint
- GetParentKey() : Script::Task
- GetParentNodeIndex() : IO::BXmlLoaderUtil
- `GetParentWindow()` : `Base::DisplayDeviceBase`, `Graphics::Display`, `Base::DisplayDeviceBase`
- `GetParticleSystem()` : `Particles::ParticleSystemNode`
- `GetParticleSystemInstance()` : `Particles::ParticleSystemNodeInstance`
- `GetParticleVertexBuffer()` : `Direct3D9::D3D9ParticleRenderer`
- `GetPassRenderTarget()` : `Base::RenderDeviceBase`
- `GetPath()` : `IO::Assign`
- `GetPayload()` : `Net::DebugPacket`
- `GetPayloadSize()` : `Net::DebugPacket`
- `GetPhysicsEntity()` : `PhysicsFeature::PhysicsProperty`, `PhysicsFeature::ActorPhysicsProperty`
- `GetPixelFormat()` : `Base::TextureBase`, `CoreGraphics::DisplayMode`
- `GetPixelPosition()` : `Base::MouseBase`
- `GetPlaceholderResourceId()` : `Resources::ResourceMapper`
- `GetPlatform()` : `Base::SystemInfoBase`
- `GetPlayerIndex()` : `Base::GamePadBase`
- `GetPointer()` : `Jobs::JobUniformDesc`, `Jobs::JobDataDesc`
- `GetPoolForDebug()` : `Resources::PoolResourceMapper`
- `GetPoolSize()` : `Win360::Win360MemoryPool`, `OSX::OSXMemoryPool`
- `GetPort()` : `Win360::Win360IpAddress`, `Http::HttpServer`, `Net::DebugMessage`, `Net::DebugPacket`
- `GetPoseMatrix()` : `Characters::CharacterJoint`
- `GetPoseRotation()` : `Characters::CharacterJoint`
- `GetPoseScale()` : `Characters::CharacterJoint`
- `GetPoseTranslation()` : `Characters::CharacterJoint`
- `GetPosition()` : `Models::TransformNodeInstance`, `IO::MemoryStream`, `Models::TransformNode`, `CoreGraphics::MousePointer`, `CoreGraphics::TextElement`, `Models::TransformNode`, `IO::ZipFileStream`, `Models::TransformNode`, `Models::TransformNodeInstance`, `Models::TransformNode`, `IO::FileStream`, `Models::TransformNodeInstance`
- `getposition()` : `Math::transform44`
- `GetPosition()` : `Models::TransformNodeInstance`, `Visibility::ObserverContext`, `IO::Stream`
- `GetPositionAmplitude()` : `GraphicsFeature::LightFlickerUtil`
- `GetPostInfinityType()` : `CoreAnimation::AnimClip`
- `GetPreInfinityType()` : `CoreAnimation::AnimClip`
- `GetPresentParams()` : `Direct3D9::D3D9RenderDevice`
- `GetPrimitiveGroup()` : `Base::RenderDeviceBase, Direct3D9::D3D9ParticleRenderer, Base::RenderDeviceBase`
- `GetPrimitiveGroupAtIndex()` : `Base::MeshBase`
- `GetPrimitiveGroupIndex()` : `Models::ShapeNode, Particles::ParticleSystemNode, Models::ShapeNode`
- `GetPrimitiveTopology()` : `CoreGraphics::PrimitiveGroup`
- `GetPriority()` : `Resources::ManagedResource, OSX::OSXThread, Win360::Win360Thread, Resources::LoadingResource, Win360::Win360Thread, Resources::ManagedResource, Win360::Win360Thread, Resources::LoadingResource, Resources::ManagedResource`
- `GetPrivateBuffer()` : `Base::JobBase`
- `GetPrivateBufferSize()` : `Base::JobBase`
- `GetProfileDirectory()` : `BaseGameFeature::UserProfile`
- `GetProfileRootDirectory()` : `BaseGameFeature::UserProfile`
- `GetProgramsDirectory()` : `OSX::OSXFSWrapper, Win360::Win360FSWrapper`
- `GetProgressResource()` : `BaseGameFeature::LoaderServer`
- `GetProgressText()` : `BaseGameFeature::LoaderServer`
- `GetProjectionMatrix()` : `Visibility::ObserverContext`
- `GetProjMapUvOffsetAndScale()` : `Graphics::AbstractLightEntity, Lighting::InternalAbstractLightEntity`
- `GetProjTransform()` : `Base::TransformDeviceBase, Shared::CameraSettings, Graphics::CameraEntity, InternalGraphics::InternalCameraEntity, Lighting::InternalAbstractLightEntity`
- `GetPSSMSplitDistances()` : `Lighting::SM30ShadowServer`
- `GetPSSMSplitLightProjTransforms()` : `Lighting::SM30ShadowServer`
- `GetPtr()` : `Util::Blob`
- `GetRawBuffer()` : `Net::DebugPacket`
- `GetRawPointer()` : `IO::MemoryStream`
- `GetRecvBufSize()` : `Win360::Win360Socket`
- `GetRecvStream()` : `Net::MessageClientConnection`
Net::StdTcpClientConnection, Net::StdTcpClient, Net::StdTcpClientConnection, Net::MessageClient

- GetRefCount(): Core::RefCounted
- GetRenderCount(): Resources::ManagedResource
- GetRenderDebug(): InternalGraphics::InternalGraphicsServer
- GetRenderModules(): Graphics::GraphicsServer
- GetRenderTarget(): Frame::FramePassBase, InternalGraphics::InternalView, Base::MultipleRenderTargetBase
- GetRenderTargetByIndex(): Frame::FrameShader
- GetRenderTargetByName(): Frame::FrameShader
- GetRequestHandlers(): Http::HttpServer
- GetRequestURI(): Http::HttpRequestReader
- GetResolveDepthTextureResourceId(): Base::RenderTargetBase
- GetResolveRect(): Base::RenderTargetBase
- GetResolveTargetCpuAccess(): Base::RenderTargetBase
- GetResolveTexture(): Base::RenderTargetBase
- GetResolveTextureHeight(): Base::RenderTargetBase
- GetResolveTextureResourceId(): Base::RenderTargetBase
- GetResolveTextureWidth(): Base::RenderTargetBase
- GetResourcesByType(): Resources::ResourceManager
- GetResourceState(): Models::TransformNode,
Models::ShapeNode, Models::ModelNode, Characters::CharacterNode, Models::StateNode, Particles::ParticleSystemNode
- GetResourceStreamingLevelOfDetail(): Resources::ManagedResource
- GetResourceType(): Resources::ManagedResource, Resources::SimpleResourceMapper, Resources::ManagedResource, Resources::ResourceMapper, Resources::ManagedResource
- GetResponseAtIndex(): Script::Dialog
- GetResponseContentStream(): Http::HttpRequest
- GetResponsePassedStates(): Script::Dialog, Script::DialogManager
- GetResponses(): Script::DialogManager
- GetResponseSoundIDs(): Script::DialogManager, Script::Dialog
- GetResponseSpeakers(): Script::DialogManager, Script::Dialog
- GetResponseTexts(): Script::Dialog
- GetResult(): Animation::AnimSequencer, Conditions::Condition, Util::Crc, Conditions::Condition
- GetReturnCode(): App::Application
- GetReUseAddr(): Win360::Win360Socket
- GetReuseTexture(): Resources::D3D9TextureStreamer
- GetRevision(): CoreGraphics::AdapterInfo
- GetRootDirectory(): Interface::InterfaceBase, Core::CoreServer
- GetRootLocation(): Http::HttpRequestHandler
- GetRootNode(): Models::Model
- GetRootNodeInstance(): Models::ModelInstance
- GetRootNodeOffsetMatrix(): Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- GetRootNodePath(): InternalGraphics::InternalModelEntity, Graphics::ModelEntity
- GetRotate(): Models::TransformNodeInstance
- getrotate(): Math::transform44
- GetRotate(): Models::TransformNodeInstance
- getrotatepivot(): Math::transform44
- `GetRotatePivot()` : `Models::TransformNode`, `Models::TransformNodeInstance`, `Models::TransformNode`
- `GetRotation()` : `Models::TransformNode`
- `getrow0()` : `Math::matrix44`
- `getrow1()` : `Math::matrix44`
- `getrow2()` : `Math::matrix44`
- `getrow3()` : `Math::matrix44`
- `GetRowIndexByName()` : `Util::SparseTable< TYPE >`
- `GetRTPlugins()` : `RenderModules::RTPluginRegistry`
- `GetSafeRLEBufferSize()` : `Util::RunLengthCodec`
- `GetSample()` : `Debug::DebugCounter`, `Debug::DebugTimer`
- `GetSampleBuffer()` : `Particles::EnvelopeSampleBuffer`
- `GetSampleCountsPointer()` : `CoreAnimation::AnimSampleBuffer`
- `GetSamplesPointer()` : `CoreAnimation::AnimSampleBuffer`
- `GetSaveActionsFlag()` : `Actions::SequenceAction`
- `GetSaveGame()` : `BaseGameFeature::GameStateHandler`
- `GetSaveGameDirectory()` : `BaseGameFeature::UserProfile`
- `GetSaveGamePath()` : `BaseGameFeature::UserProfile`
- `GetSaver()` : `Resources::Resource`
- `GetScale()` : `Models::TransformNode`, `Models::TransformNodeInstance`, `Models::TransformNode`
- `getscale()` : `Math::transform44`
- `GetScale()` : `Models::TransformNodeInstance`, `Models::TransformNode`, `Models::TransformNodeInstance`, `Models::TransformNode`
- `getscalepivot()` : `Math::transform44`
- `GetScaledFrameTime()` : `FrameSync::FrameSyncTimer`
- `GetScaledTime()` : `FrameSync::FrameSyncTimer`
- `GetScalePivot()` : `Models::TransformNodeInstance`, `Models::TransformNode`, `Models::TransformNodeInstance`, `Models::TransformNode`
- `getscalepivot()` : `Math::transform44`
- `GetScalePivot()` : `Models::TransformNode`, `Models::TransformNodeInstance`
- `GetScratchSize()` : `Jobs::JobUniformDesc`
- `GetScreenPosition()` : `Base::MouseBase`
- `GetScreenshotFilepath()` : `BaseGameFeature::LoaderServer`
- `GetSecond()` : `Base::CalendarTimeBase`
- `GetSemantic()` : `Base::ShaderVariableBase`
- GetSendBufSize(): **Win360::Win360Socket**
- GetSendStream(): **Net::MessageClientConnection**, **Net::StdTcpClient**, **Net::StdTcpClientConnection**, **Net::StdTcpClient**, **Net::StdTcpClientConnection**, **Net::MessageClient**
- GetServerAddress(): **Net::StdTcpClient**
- GetSetupMode(): **BaseGameFeature::GameStateHandler**
- GetShader(): **Models::StateNode** , **Frame::FramePassBase** , **Models::StateNode** , **Frame::FrameBatch** , **Frame::FramePassBase** , **Models::StateNode**
- GetShaderFeatures(): **Frame::FrameBatch**
- GetShaderInstance(): **Models::StateNode**
- GetShaderParameter(): **Models::StateNode**
- GetShaderVariable(): **Base::ShaderVariableInstanceBase**
- GetShaderVariableInstance(): **Models::StateNodeInstance**
- GetShadowBufferUvOffsetAndScale(): **Lighting::InternalAbstractLightEntity**
- GetShadowIntensity(): **Lighting::InternalAbstractLightEntity**
- GetShadowInvLightProjTransform(): **Lighting::InternalAbstractLightEntity**
- GetShadowInvTransform(): **Lighting::InternalAbstractLightEntity**
- GetShadowProjTransform(): **Lighting::InternalAbstractLightEntity**
- GetShapes(): **PhysicsFeature::EnvironmentCollideProperty**
- GetShapeType(): **CoreGraphics::RenderShape**
- GetSharedVariableByIndex(): **Base::ShaderServerBase**
- GetSharedVariableBySemantic(): **Base::ShaderServerBase**
- GetShortResponseTexts(): **Script::Dialog**, **Script::DialogManager**
- GetShortText(): **Script::DialogTake**
- GetSize(): **IO::FileStream**, **IO::MemoryStream**, **CoreGraphics::MousePointer**, **Resources::ResourceDictionary::Entry**, **IO::Stream**, **IO::ZipFileStream**
- GetSkin(): **Characters::CharacterSkinSet**, **Characters::CharacterSkinLibrary**
- GetSkinByName(): **Characters::CharacterSkinLibrary**
- GetSkinIndexByName(): **Characters::CharacterSkinLibrary**
- GetSkinList() : Characters::CharacterSkinLibrary
- GetSkinListByName() : Characters::CharacterSkinLibrary
- GetSkinListIndexByName() : Characters::CharacterSkinLibrary
- GetSkinMatrixArray() : Characters::CharacterSkeletonInstance
- GetSkinningTechnique() : Characters::CharacterServer, Base::SkinnedMeshRendererBase, Win32::Win32SkinnedMeshRenderer
- GetSkinNodeInstanceByIndex() : Characters::CharacterSkinSet
- GetSkinNodeInstanceByName() : Characters::CharacterSkinSet
- GetSkins() : Characters::CharacterSkinList
- GetSkippedMips() : Base::TextureBase
- GetSliceIndex() : Jobs::TPJobSlice
- GetSliceSize() : Jobs::JobDataDesc
- GetSortingMode() : Frame::FrameBatch
- GetSound() : Script::DialogTake
- GetSource() : Script::InfoLog
- GetSpeaker() : Script::DialogManager, Script::DialogDesc, Script::Dialog, Script::DialogTake
- GetSplitDistances() : Lighting::PSSMUtil
- GetSplitLightProjTransform() : Lighting::PSSMUtil
- GetSplitLightProjTransforms() : Lighting::PSSMUtil
- GetSplitLightTransform() : Lighting::PSSMUtil
- GetSplitProjTransform() : Lighting::PSSMUtil
- GetStackSize() : Win360::Win360Thread, OSX::OSXThread
- GetStageByName() : InternalGraphics::InternalGraphicsServer, Graphics::GraphicsServer
- GetStageName() : Graphics::View
- GetStartKeyIndex() : CoreAnimation::AnimClip
- GetStartTime() : Animation::AnimJob, CoreAnimation::AnimClip, Animation::AnimJob
- GetStateAsInputEvents() : Base::GamePadBase
- GetStateHandlerAt() : App::GameApplication
- GetStaticKey() : CoreAnimation::AnimCurve
- GetStatus() : Http::HttpRequest
- GetStatusCode() : Http::HttpResponseReader
- GetStream() : IO::StreamReader, IO::StreamWriter, IO::StreamReader, Base::StreamTextureSaverBase, IO::StreamWriter, Base::StreamTextureSaverBase, IO::StreamWriter, IO::StreamReader, IO::StreamWriter, IO::StreamReader, Base::StreamTextureSaverBase, Net::DebugMessage, IO::StreamWriter, IO::StreamReader, IO::StreamWriter, IO::StreamReader
- GetStreamByteOrder() : IO::BinaryReader, IO::BinaryWriter
- GetStreamClassByUriScheme() : IO::SchemeRegistry
- GetStreamVertexBuffer() : Base::RenderDeviceBase
- GetStreamVertexOffset() : Base::RenderDeviceBase
- GetStride() : Jobs::TPJobCommand
- GetString() : BaseGameFeature::UserProfile, Script::ActionReader, Util::Variant, Util::CommandLineArgs, IO::BXmlReader, IO::XmlReader, BaseGameFeature::GlobalAttrsManager, Game::Entity
- GetStringArray() : Util::Variant
- GetStringAtIndex() : Util::CommandLineArgs
- getStringAttr() : Models::ModelNode
- GetSubstitution() : Script::SubstitutionManager
- GetSubSystemId() : CoreGraphics::AdapterInfo
- GetSubTasks() : Script::Task
- GetSubType() : IO::MediaType
- GetSurfaceInfo() : Resources::D3D9TextureStreamer
- GetSyncEvent() : Jobs::TPJobCommand
- GetSystemInfo() : Win32::SysFunc, OSX::SysFunc
- GetSystemTime() : Base::CalendarTimeBase, Win360::Win360CalendarTime
- GetTableIndex() : IO::ExcelXmlReader
- GetTableName() : IO::ExcelXmlReader
- GetTail() : IO::URI
- GetTakeById() : Script::Dialog
- GetTakeSoundID() : Script::DialogManager, Script::Dialog
- GetTakeSpeaker() : Script::Dialog, Script::DialogManager
- GetTargetEntityName() : Script::Task
- GetTargetResource() : Resources::LoadingResource
- GetTargetState() : FSM::Transition
- GetTaskViewed() : Script::Task
- GetTempDirectory() : Win360::Win360FSWrapper, OSX::OSXFSWrapper
- GetTemplateDataset() :
  BaseGameFeature::CategoryManager::Category
- GetTemplateTable() : BaseGameFeature::CategoryManager
- GetTemplateTableName() :
  BaseGameFeature::CategoryManager::Category
- GetText() : CoreGraphics::TextElement, Script::Task
- GetTextList() : Script::DialogTake
- GetTexture() : Resources::ManagedTexture
- GetTextureByIndex() : Frame::FrameShader
- GetTextureByName() : Frame::FrameShader
- GetTextureUnloaded() : Resources::ManagedTexture
- GetThenBlock() : Actions::IfThenElseAction
- GetThreadId() : CoreGraphics::RenderShape, CoreGraphics::TextElement
- GetTicks() : BaseGameFeature::TimeSource, FrameSync::FrameSyncTimer, BaseGameFeature::TimeSource, Win360::Win360Timer,
BaseGameFeature::TimeSource
- getTime(): Win360::Win360Timer, BaseGameFeature::TimeSource, CoreAnimation::AnimEvent, App::RenderApplication, BaseGameFeature::TimeSource, FrameSync::FrameSyncTimer, Animation::AnimSequencer, BaseGameFeature::TimeSource, App::RenderApplication, BaseGameFeature::TimeSource
- getTimeFactor(): Animation::AnimJob, BaseGameFeature::TimeSource, FrameSync::FrameSyncTimer, Animation::AnimJob
- getTimeLeft(): Actions::Action, Actions::SequenceAction, Actions::Action
- getTimeOffset(): Animation::AnimJob
- getTimeSourceByClassName(): BaseGameFeature::TimeManager
- getTimeSourceByIndex(): BaseGameFeature::TimeManager
- getTimeStamp(): Script::DialogTake
- getTitle(): Script::Task, Base::GameContentServerBase
- getTitleId(): Base::GameContentServerBase
- getToByteOrder(): System::ByteOrder
- getTopology(): CoreGraphics::RenderShape
- getTrackedCharJointInfo(): Graphics::ModelEntity
- getTrackIndex(): Animation::AnimJob
- getTransitionAtIndex(): FSM::State
- getTriggerPosition(): PhysicsFeature::TriggerProperty
- getTriggerScale(): PhysicsFeature::TriggerProperty
- getTriggerTransform(): PhysicsFeature::TriggerProperty
- getType(): Graphics::GraphicsEntity, Models::ModelNode, Script::DialogTake, InternalGraphics::InternalGraphicsEntity
, Base::TextureBase,
InternalGraphics::InternalGraphicsEntity,
Models::ModelNode, Util::Variant, Frame::FrameBatch,
Models::ModelNode,
InternalGraphics::InternalGraphicsEntity, Base::TextureBase,
Models::ModelNode,
InternalGraphics::InternalGraphicsEntity, Baset::ShaderVariableBase,
Visibility::ObserverContext, Models::ModelNode,
Base::TextureBase, InternalGraphics::InternalGraphicsEntity,
Input::InputEvent, Graphics::GraphicsEntity,
Models::ModelNode, Graphics::GraphicsEntity,
InternalGraphics::InternalGraphicsEntity, IO::MediaType,
Models::ModelNode, Graphics::GraphicsEntity,
Models::ModelNode,
InternalGraphics::InternalGraphicsEntity

- GetTypeRef() : Visibility::ObserverContext
- GetUniformDesc() : Base::JobBase
- GetUniqueId() : Game::Entity
- GetUpVector() : BaseGameFeature::EnvQueryManager
- GetURI() : IO::Stream, Http::HttpRequest, IO::Stream,
Resources::ResourceDictionary, IO::Stream,
IO::ArchiveBase
- GetUsage() : Base::ResourceBase,
CoreGraphics::MemoryMeshLoader, Base::ResourceBase,
Win360::D3D9StreamMeshLoader, Base::ResourceBase,
Win360::D3D9StreamMeshLoader, Base::ResourceBase
- GetUseCount() : Resources::Resource
- GetUsedMemory() : Resources::PoolResourceMapper
- GetUserDirectory() : Win360::Win360FSWrapper,
OSX::OSXFSWrapper
- GetUserProfile() : BaseGameFeature::LoaderServer
- GetVariableByIndex() : Frame::FrameBatch,
Base::ShaderInstanceBase, Frame::FramePassBase,
Base::ShaderInstanceBase, Frame::FramePassBase,
Base::ShaderInstanceBase, Frame::FramePassBase
- GetVariableByName() : Base::ShaderInstanceBase
- GetVariableBySemantic() : Base::ShaderInstanceBase
- GetVariationByFeatureMask() : Base::ShaderInstanceBase
- GetVariationByIndex() : Base::ShaderInstanceBase
- GetVariationCharJointComponents() : Characters::CharacterVariationSet
- GetVector() : Script::ActionReader
- GetVendorId() : CoreGraphics::AdapterInfo
- GetVersion() : Base::GameContentServerBase
- GetVertexBuffer() : Base::MeshBase
- GetVertexBufferByteSize() : Base::VertexBufferBase
- GetVertexComponents() : Base::VertexBufferBase
- GetVectorData() : CoreGraphics::RenderShape
- GetVectorLayout() : Direct3D9::D3D9ParticleRenderer, Base::VertexBufferBase, Base::RenderDeviceBase, Base::VertexBufferBase, Direct3D9::D3D9ParticleRenderer, Base::RenderDeviceBase, Base::VertexBufferBase
- GetVectorWidth() : CoreGraphics::RenderShape
- GetView() : Graphics::CameraEntity
- GetViewByName() : Graphics::GraphicsServer, InternalGraphics::InternalGraphicsServer
- GetViewClass() : Graphics::View
- GetViewFrustum() : InternalGraphics::InternalCameraEntity, Graphics::CameraEntity, Shared::CameraSettings
- GetViewProjTransform() : Base::TransformDeviceBase, Graphics::CameraEntity, Shared::CameraSettings, Base::TransformDeviceBase, InternalGraphics::InternalCameraEntity
- GetViewTransform() : Shared::CameraSettings, Base::TransformDeviceBase, Graphics::CameraEntity, Base::TransformDeviceBase, InternalGraphics::InternalCameraEntity
- GetVisibilityChecker() : InternalGraphics::InternalStage
- GetVisibilityContexts() : Visibility::VisibilityContainer
- GetVisibleEntities() : Visibility::VisibilityQuery
- GetVisibleFrameId() : Visibility::VisibilityContext
- GetVisibleModelNodeInstances() : Models::ModelNode, Models::VisResolver
- GetVisibleModelNodes() : Models::VisResolver
- GetVisibleModels() : Models::VisResolver
- GetVisualizeMipMaps() : Base::RenderDeviceBase
- GetWaitTimeout() : Messaging::BlockingHandlerThread
- GetWeekday() : Base::CalendarTimeBase
- GetWeight() : Animation::AnimEventInfo
- GetWidth() : Base::RenderTargetBase,
  CoreGraphics::DisplayMode, Base::TextureBase,
  Base::RenderTargetBase, Base::TextureBase
- GetWindowTitle() : Base::DisplayDeviceBase,
  Graphics::DisplaySettings, Base::DisplayDeviceBase
- GetXPos() : CoreGraphics::DisplayMode
- GetYear() : Base::CalendarTimeBase
- GetYPos() : CoreGraphics::DisplayMode
- GetZFar() : Shared::CameraSettings
- GetZNear() : Shared::CameraSettings
- GlobalAttrsManager() :
  BaseGameFeature::GlobalAttrsManager
- GlobalLightEntity() : Graphics::GlobalLightEntity
- GlobalStringAtomTable() : Util::GlobalStringAtomTable
- GraphicsEntity() : Graphics::GraphicsEntity
- GraphicsEntityShared() : Shared::GraphicsEntityShared
- GraphicsHandler() : Graphics::GraphicsHandler
- GraphicsInterface() : Graphics::GraphicsInterface
- GraphicsPageHandler() : Debug::GraphicsPageHandler
- GraphicsProperty() : GraphicsFeature::GraphicsProperty
- GraphicsServer() : Graphics::GraphicsServer
- greater3_all() : Math::float4
- greater3_any() : Math::float4
- greater4_all() : Math::float4
- greater4_any() : Math::float4
- greaterequal3_all() : Math::float4
- greaterequal3_any() : Math::float4
- greaterequal4_all() : Math::float4
- greaterequal4_any() : Math::float4
- gt() : Math::float2
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Enumerator
- Related Functions

- _
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
- h -

- Handle() : **Messaging::StaticMessageHandler**
- HandleAnimEvents() : **Animation::AnimEventServer**
- HandleCameraDistanceChange() : **GraphicsFeature::ChaseCameraProperty**
- HandleCameraOrbit() : **GraphicsFeature::ChaseCameraProperty**
- HandleCameraReset() : **GraphicsFeature::ChaseCameraProperty**
- HandleCharacterAnimDrivenMotion() : **InternalGraphics::InternalModelEntity**
- HandleCharacterAnimEvents() : **InternalGraphics::InternalModelEntity**
- HandleCommand() : **Commands::CreateEntityCommand**
- Handled() : **Messaging::Message**
- HandleDefaultEntity() : **Commands::CreateEntityCommand**
- HandleDeferredMessages() : **InternalGraphics::InternalGraphicsEntity**
- HandleDelayedJobs() : **BaseGameFeature::EntityManager**
- HandleEvent() : **Animation::AnimEventHandlerBase**, **CoreGraphics::DisplayEventHandler**, **CoreGraphics::RenderEventHandler**, **CoreGraphics::ThreadSafeDisplayEventHandler**, **CoreGraphics::ThreadSafeRenderEventHandler**, **Win32::Win32InputDisplayEventHandler**
- HandleLeftMouseBtnDown() : PhysicsFeature::MouseGripperProperty
- HandleLeftMouseBtnUp() : PhysicsFeature::MouseGripperProperty
- HandleMoveDirection() : PhysicsFeature::ActorPhysicsProperty
- HandleMoveFollow() : PhysicsFeature::ActorPhysicsProperty
- HandleMoveGoto() : PhysicsFeature::ActorPhysicsProperty
- HandleMoveRotate() : PhysicsFeature::ActorPhysicsProperty
- HandleMoveTurn() : **PhysicsFeature::ActorPhysicsProperty**
- HandlePendingEvents() :
  - **CoreGraphics::ThreadSafeDisplayEventHandler** ,
  - **CoreGraphics::ThreadSafeRenderEventHandler** ,
  - **CoreGraphics::ThreadSafeDisplayEventHandler**
- HandlePendingRequests() : **Http::HttpRequestHandler**, **Http::HttpServerProxy**, **Http::HttpRequestHandler**
- Handler() : **Messaging::Handler**
- HandleRequest() : **Debug::StreamingTexturePageHandler**, **Debug::HelloWorldRequestHandler**, **Debug::ObjectInspectorHandler**, **Debug::CorePageHandler**, **Debug::DebugPageHandler**, **Debug::GraphicsPageHandler**, **Debug::MemoryPageHandler**, **Debug::SvgTestPageHandler**, **Debug::DisplayPageHandler**, **Debug::MeshPageHandler**, **Debug::ShaderPageHandler**, **Debug::TexturePageHandler**, **Debug::ConsolePageHandler**, **Debug::ThreadPageHandler**, **Debug::IoPageHandler**, **Http::DefaultHttpRequestHandler**, **Http::HttpRequestHandler**
- HandlerThreadBase() : **Messaging::HandlerThreadBase**
- HandleSetTransform() :
  - **PhysicsFeature::ActorPhysicsProperty**
- HandleTrackedJoints() : **InternalGraphics::InternalModelEntity**
- HasActionScript() : **Script::ScriptManager**
- HasActiveEntities() : **BaseGameFeature::EntityManager**
- HasArg() : **Util::CommandLineArgs**
- HasAssign() : **IO::AssignRegistry**
- HasAttr() : **BaseGameFeature::UserProfile**, **IO::XmlReader**, **BaseGameFeature::GlobalAttrsManager**, **Game::Entity**, **IO::BXmlReader**
- HasCategory() : **BaseGameFeature::CategoryManager**, **CoreAnimation::AnimEvent**
- HasCharacter() : **InternalGraphics::InternalModelEntity**
- HasChild() : **Models::ModelNodeInstance**, **Models::ModelNode**, **Models::ModelNodeInstance** , **Models::ModelNode**, **Models::ModelNodeInstance** , **Models::ModelNode**, **Models::ModelNodeInstance** , **Models::ModelNode**, **Models::ModelNodeInstance** , **Models::ModelNode**
- HasClip() : **CoreAnimation::AnimResource**
HasColumn() : IO::ExcelXmlReader , Util::SparseTable< TYPE >
HasComponent() : Base::VertexLayoutBase
HasCondition() : Script::DialogTake
HasContent() : IO::XmlReader
HasCPUResolveTexture() : Base::RenderTargetBase
HasData() : Net::DebugPacket
HasDepthResolveTexture() : Base::RenderTargetBase
HasDepthStencilBuffer() : Base::RenderTargetBase
HasDialogs() : Script::DialogManager
HasElseBlock() : Actions::IfThenElseAction
HasEntity() : Conditions::Condition , Game::Property ,
Conditions::Condition , Game::Property ,
Conditions::Condition , Game::Property ,
Conditions::Condition , Game::Property , Actions::Action ,
InternalGraphics::InternalGraphicsServer , Actions::Action ,
Game::Property
HasEntry() : Resources::ResourceDictionary
HasEvent() : CoreAnimation::AnimClip
HasFocus() : GraphicsFeature::CameraProperty
HasFramePassBase() : Frame::FrameShader
HashCode() : Util::String , Util::Blob , Win32::Win32Guid ,
OSX::OSXGuid
HasTable() : Util::HashTable< KEYTYPE, VALUETYPE >
HasIndexBuffer() : Base::MeshBase
HasInfo() : Script::InfoLog
HasInfoInSection() : Script::InfoLog
HasInput() : IO::ConsoleHandler , OSX::OSXConsoleHandler ,
IO::ConsoleHandler , IO::Console ,
Win32::Win32ConsoleHandler , IO::ConsoleHandler
HasInstanceDataset() :
BaseGameFeature::CategoryManager::Category
HasInstanceTable() : BaseGameFeature::CategoryManager
HasLightEntity() : GraphicsFeature::LightFlickerUtil
HasManagedModel() : Models::ModelServer
HasManagedResource() : Resources::ResourceManager
HasMapper() : Resources::ResourceManager
HasMessages() : Net::TcpMessageCodec
HasMouseIntersection() :
BaseGameFeature::EnvQueryManager
  HasMultipleRenderTarget() : Frame::FrameShader
  HasNextTakeLayer() : Script::Dialog
  HasNode() : IO::XmlReader, IO::BXmlReader
  HasOpenConditions() : Script::Task
  HasParent() : Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Util::SimpleTree< VALUETYPE >::Node
  HasParentJoint() : Characters::CharacterJoint
  HasPassRenderTarget() : Base::RenderDeviceBase
  HasPort() : Messaging::Dispatcher
  HasRecvData() : Win360::Win360Socket
  HasRenderDebug() : Game::FeatureUnit
  HasRenderTarget() : Frame::FrameShader, InternalGraphics::InternalView
  HasResolveTexture() : Base::RenderTargetBase
  HasResource() : Resources::ResourceManager
  HasRow() : Util::SparseTable<TYPE>
  HasShader() : Base::ShaderServerBase
  HasShaderVariableInstance() : Models::StateNodeInstance
  HasShapes() : PhysicsFeature::EnvironmentCollideProperty
  HasSharedVariableBySemantic() : Base::ShaderServerBase
  HasSkin() : Characters::CharacterSkinLibrary, Characters::CharacterSkinSet
  HasSkinList() : Characters::CharacterSkinLibrary
  HasStage() : InternalGraphics::InternalGraphicsServer, Graphics::GraphicsServer
  HasStarted() : Game::GameServer
  HasStream() : IO::StreamReader, IO::StreamWriter, IO::StreamReader, IO::StreamWriter, IO::StreamReader, IO::StreamWriter, IO::StreamReader, IO::StreamWriter, IO::StreamReader, IO::StreamWriter
  HasStringAttr() : Models::ModelNode
  HasSubTasks() : Script::Task
  HasTaskStatusChanged() : Script::Task
- HasTemplateDataset(): BaseGameFeature::CategoryManager::Category
- HasTemplateTable(): BaseGameFeature::CategoryManager
- HasTexture(): Frame::FrameShader
- HasVariableByName(): Base::ShaderInstanceBase
- HasVariableBySemantic(): Base::ShaderInstanceBase
- HasVariation(): Base::ShaderInstanceBase
- HasVertexBuffer(): Base::MeshBase
- HasView(): InternalGraphics::InternalGraphicsServer, Graphics::GraphicsServer
- height(): Math::rectangle< TYPE >
- Height(): Util::FixedTable< TYPE >
- HelloWorldRequestHandler(): Debug::HelloWorldRequestHandler
- hermite(): Math::float4
- HistoryConsoleHandler(): IO::HistoryConsoleHandler
- HoldResources(): Resources::ResourceManager
- HorizontalRule(): Http::HtmlPageWriter
- Host(): IO::URI
- HtmlPageWriter(): Http::HtmlPageWriter
- HttpInterface(): Http::HttpInterface
- HttpMessageHandler(): Http::HttpMessageHandler
- HttpRequest(): Http::HttpRequest
- HttpRequestHandler(): Http::HttpRequestHandler
- HttpRequestReader(): Http::HttpRequestReader
- HttpRequestWriter(): Http::HttpRequestWriter
- HttpResponseReader(): Http::HttpResponseReader
- HttpServer(): Http::HttpServer
- HttpServerProxy(): Http::HttpServerProxy
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q
- i -

- r
- s
- t
- u
- v
- w
- x
- y
- z
- ~

- Id() : Messaging::Id
- identity() : Math::matrix44, Math::quaternion
- IfThenElseAction() : Actions::IfThenElseAction
- InbetweenTicks() : CoreAnimation::AnimUtil
- Incr() : Debug::DebugCounter
- IncrClientCount() : Resources::ManagedResource
- Increment() : OSX::OSXInterlocked, Win360::Win360Interlocked
- IncrUseCount() : Resources::Resource
- IndexBufferBase() : Base::IndexBufferBase
- IndexTypeAsD3D9Format() : Win360::D3D9Types
- InfoLog() : Script::InfoLog
- Init() : Actions::ActionList, Actions::Action
- InitNetwork() : Win360::Win360Socket
- InputEvent() : Input::InputEvent
- InputHandler() : Input::InputHandler
- InputServer() : Input::InputServer
- InputServerBase() : Base::InputServerBase
- InputTimeSource() : BaseGameFeature::InputTimeSource
- Insert() : Util::Array<TYPE>, Threading::SafePriorityQueue<PRITYPE, TYPE>, Util::Array<TYPE>, Util::SimpleTree<VALUETYPE>::Node
- InsertAtEndOfIdenticalRange() : Util::Array<TYPE>
- InsertSorted() : Util::Array<TYPE>
- InsertVisibilityContainer() : Visibility::VisibilityClusterSystem, Visibility::VisibilitySystemBase, Visibility::VisibilityBoxSystem, Visibility::VisibilitySystemBase
- InsertVisibilityContext() : Visibility::VisibilityBoxSystem, Visibility::VisibilityClusterSystem, Visibility::VisibilityQuadtree, Visibility::VisibilitySystemBase
- inside() : Math::frustum, Math::rectangle< TYPE >, Math::sphere
- Instance() : Core::Factory
- InterfaceBase() : Interface::InterfaceBase
- InterfaceHandlerBase() : Interface::InterfaceHandlerBase
- InternalAbstractLightEntity() : Lighting::InternalAbstractLightEntity
- InternalCameraEntity() : InternalGraphics::InternalCameraEntity
- InternalGlobalLightEntity() : Lighting::InternalGlobalLightEntity
- InternalModelEntity() : InternalGraphics::InternalModelEntity
- InternalPointLightEntity() : Lighting::InternalPointLightEntity
- InternalSpotLightEntity() : Lighting::InternalSpotLightEntity
- InternalStage() : InternalGraphics::InternalStage
- InternalView() : InternalGraphics::InternalView
- intersect() : Math::line
- intersect_sweep() : Math::sphere
- intersectline() : Math::plane
- intersects() : Math::sphere, Math::bbox
- Invalidate() : Threading::ObjectRef
- inverse() : Math::matrix44, Math::quaternion
- Invoke() : Messaging::DelegateTable
- IoInterfaceHandler() : IO::IoInterfaceHandler
- IoPageHandler() : Debug::IoPageHandler
- IoServer() : IO::IoServer
- IsA() : Core::RefCounted
- IsActive() : Game::Manager, Game::Property, Game::FeatureUnit, Game::Manager, Game::Property,
Game::Manager, Game::FeatureUnit, Game::Property, Animation::AnimJob, Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property, Game::Manager, Game::FeatureUnit, Script::Dialog, Game::Property, InternalGraphics::InternalGraphicsEntity, CoreAnimation::AnimCurve, InternalGraphics::InternalGraphicsEntity, Animation::AnimJob, Game::Property, Game::Manager, Game::Property, Game::FeatureUnit, InternalGraphics::InternalGraphicsEntity, Game::Manager

- IsAlNum() : Util::String
- IsAlpha() : Util::String
- IsAlwaysOnTop() : Base::DisplayDeviceBase, Graphics::DisplaySettings, Base::DisplayDeviceBase
- IsAnimDrivenMotionEnabled() :
  Characters::CharacterAnimationController
- IsAnimDrivenMotionTrackingEnabled() : Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- IsAnimEventTrackingEnabled() : Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- IsArchiveFileSystemEnabled() : IO::IoServer
- IsArchiveMounted() : IO::IoServer
- IsAsyncEnabled() : Resources::Resource, Resources::ResourceMapper, Resources::Resource, Resources::ResourceMapper, Resources::Resource
- IsAttached() : Input::InputHandler
- IsAttachedToModel() : Models::ModelNode
- IsAttachedToResource() : Resources::ResourceLoader, Resources::ResourceSaver, Resources::ResourceLoader, Resources::ResourceSaver, Resources::ResourceLoader
- IsAttachedToResourceManager() :
  Resources::ResourceMapper
- IsAttachedToSequencer() : Animation::AnimJob
- IsAttachedToServer() : InternalGraphics::InternalStage, InternalGraphics::InternalView
- IsAttachedToStage() : InternalGraphics::InternalGraphicsEntity
- IsAttachedToView() : Graphics::CameraEntity, InternalGraphics::InternalCameraEntity
- IsAutoManaged() : Resources::ManagedResource
- IsBlocking() : Net::StdTcpClient
- IsBound() : Win360::Win360Socket
- IsCapturing() : Input::InputHandler
- IsCharJointDataValid() : Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- IsCharJointTrackingEnabled() : Graphics::ModelEntity, InternalGraphics::InternalModelEntity
- IsConnected() : Base::GamePadBase, Net::StdTcpClientConnection, Net::StdTcpClient, Net::StdTcpClientConnection, Win360::Win360Socket, Net::StdTcpClient, Net::StdTcpClientConnection, Win360::Win360Socket, Base::GamePadBase
- IsConversation() : Script::DialogDesc
- IsDebugHudEnabled() : Animation::AnimSequencer
- IsDefaultRenderTarget() : Base::RenderTargetBase
- IsDefaultView() : Graphics::View
- IsDeferred() : Messaging::Message
- IsDerivedFrom() : Core::Rtti
- IsDeviceName() : OSX::OSXFSWrapper, Win360::Win360FSWrapper
- IsDialogActive() : Script::DialogManager
- IsDialogLoaded() : Script::DialogManager
- IsDigit() : Util::String
- isdirty() : Math::transform44
- IsDisplayModeSwitchEnabled() : Base::DisplayDeviceBase, Graphics::DisplaySettings, Base::DisplayDeviceBase
- IsEmpty() : Util::Array<TYPE>, Threading::SafeQueue<TYPE>, Threading::SafePriorityQueue<PRITYPE, TYPE>, IO::URI, Util::Array<TYPE>, Util::List<TYPE>, Util::Dictionary<KEYTYPE, VALUETYPE>, Util::FixedArray<TYPE>, Util::HashTable<KEYTYPE, VALUETYPE>, Util::List<TYPE>, Util::Queue<TYPE>, Util::RingBuffer<TYPE>, Util::SimpleTree<VALUETYPE>::Node, Util::Stack<TYPE>, Util::String
- IsEnabled() : GraphicsFeature::LightFlickerUtil, PhysicsFeature::PhysicsProperty
- IsEntityInDelayedJobs() : BaseGameFeature::EntityManager
- IsExpired() : Animation::AnimJob
- IsFeatureAttached() : Game::GameServer
- IsFinalPacket() : **Net::DebugPacket**
- IsFinished() : **Visibility::VisibilityQuery**
- IsFixedFrameTime() : **FrameSync::FrameSyncHandlerThread**
- IsFullscreen() : **Base::DisplayDeviceBase**, **Graphics::DisplaySettings**, **Base::DisplayDeviceBase**
- IsGotoActive() : **PhysicsFeature::ActorPhysicsProperty**
- isidentity() : **Math::matrix44**, **Math::quaternion**
- IsInAttach() : **Base::ParticleRendererBase**
- IsInBeginFrame() : **Base::RenderDeviceBase**
- IsInfinite() : **Animation::AnimJob**
- IsInFocus() : **BaseGameFeature::EntityManager**
- IsInstanceOf() : **Core::RefCounted**
- IsInViewSpace() : **Models::TransformNode**, **Models::TransformNodeInstance**, **Models::TransformNode**, **Models::TransformNodeInstance**, **Models::TransformNode**
- IsLoaded() : **BaseGameFeature::UserProfile**, **Resources::Resource**, **Script::Dialog**, **Resources::Resource**, **Script::Task**, **Resources::Resource**
- IsLoading() : **BaseGameFeature::EntityLoaderBase**
- IsLocked() : **Resources::Resource**, **Script::DialogDesc**, **Resources::Resource**
- IsLower() : **Util::String**
- IsMapped() : **IO::Stream**, **CoreAnimation::AnimKeyBuffer**, **IO::Stream**
- IsMarkedForRemove() : **InternalGraphics::InternalGraphicsEntity**
- IsMemoryMappingEnabled() : **IO::BinaryReader**, **IO::BinaryWriter**
- IsNetworkInitialized() : **Win360::Win360Socket**
- IsNull() : **Util::BitField<NUMBITS>**
- IsObjectRefValid() : **Graphics::GraphicsEntity**
App::Application, IO::ConsoleHandler, IO::StreamReader, Net::StdTcpServer, Frame::FrameServer, IO::StreamReader, IO::Stream, IO::StreamReader, Base::RenderDeviceBase, Lighting::LightServerBase, Base::TextRendererBase, Base::VertexLayoutServerBase, IO::Stream, IO::StreamReader, Base::ShaderServerBase, Base::ShapeRendererBase, IO::StreamWriter, IO::StreamReader, Base::RenderDeviceBase, Http::HttpServerProxy, Win360::Win360Socket, Debug::DebugServer, IO::ConsoleHandler, Base::VertexLayoutServerBase, Base::RenderDeviceBase, Visibility::VisibilitySystemBase, Messaging::AsyncPort, Messaging::Handler, Lighting::LightServerBase, IO::StreamReader, IO::StreamWriter, IO::Console, BaseGameFeature::LoaderServer, Messaging::Handler, Messaging::AsyncPort, IO::Stream, Messaging::AsyncPort, Base::InputServerBase, Messaging::Handler, IO::ConsoleHandler, IO::StreamWriter, Base::DisplayDeviceBase, Visibility::VisibilitySystemBase, IO::ConsoleHandler, Models::VisResolver, Base::DisplayDeviceBase, Base::ShaderServerBase, IO::StreamWriter, Base::TransformDeviceBase, Resources::ResourceManager, Base::ShapeRendererBase, InternalGraphics::AttachmentServer, InternalGraphics::InternalGraphicsServer, App::Application, Lighting::LightServerBase, Animation::AnimEventServer, Net::StdTcpServer, Lighting::LightServerBase, IO::ConsoleHandler, Base::TransformDeviceBase, IO::StreamWriter, Base::TransformDeviceBase, IO::Stream, IO::StreamWriter, Messaging::Handler, Base::TextRendererBase, IO::StreamReader, Visibility::VisibilitySystemBase, IO::StreamWriter, Messaging::AsyncPort, Core::CoreServer, IO::StreamWriter, App::Application, Messaging::Handler, Base::ShaderServerBase, Base::DisplayDeviceBase, Base::ShapeRendererBase, Messaging::Handler, Base::DisplayDeviceBase

- IsOrthogonal() : Shared::CameraSettings
- IsPaused() : BaseGameFeature::TimeSource
- IsPending() : Animation::AnimJob, Resources::Resource, Animation::AnimJob, Resources::Resource
- IsPerfHUDEnabled() : Debug::DebugGraphicsHandler
- IsPerspective() : Shared::CameraSettings
- IsPlaceholder() : Resources::ManagedResource
- ispointinside() : Math::matrix44
- IsPoolBlock() : OSX::OSXMemoryPool, Win360::Win360MemoryPool
- IsQuitRequested() : Base::InputServerBase, App::RenderApplication, Base::InputServerBase, App::RenderApplication, Base::InputServerBase, Game::GameServer
- IsReadOnly() : OSX::OSXFSWrapper, Win360::Win360FSWrapper, IO::IoServer
- IsResolved() : Models::VisResolveContainer<Type>
- IsRunning() : Win360::Win360Thread, OSX::OSXThread, Win360::Win360Thread
- IsSignalOnEnqueueEnabled() : Threading::SafeQueue<Type>
- IsSingleThreadMode() : Http::HttpServer
- IsSorted() : Util::Array<Type>
- IsSpeaker() : Script::Dialog
- IsSpecial() : BaseGameFeature::CategoryManager::Category
- IsStatic() : CoreAnimation::AnimCurve
- IsStoppingOrExpired() : Animation::AnimJob
- IsTimeEffectActive() : BaseGameFeature::TimeManager
- IsTimeRunning() : FrameSync::FrameSyncTimer
- IsTriggerActive() : PhysicsFeature::TriggerProperty
- IsTripleBufferingEnabled() : Base::DisplayDeviceBase, Graphics::DisplaySettings
- IsUpper() : Util::String
- IsUriSchemeRegistered() : IO::SchemeRegistry
- IsValid() : Base::RenderTargetBase, Characters::CharacterSkeleton, Graphics::GraphicsEntity, Util::StringBuffer, Debug::DebugTimer, Util::String, Base::ShaderInstanceBase, Base::JobBase, Base::MouseRenderDeviceBase, Base::JobBase, Base::ParticleRendererBase, Base::GameContentServerBase, InternalGraphics::InternalGraphicsEntity,
Characters::Character, Models::ModelNodeInstance, Characters::CharacterServer, CoreAnimation::AnimSampleBuffer, Particles::ParticleSystem, Characters::CharacterSkinSet, OSX::OSXGuid, InternalGraphics::InternalGraphicsEntity, BaseGameFeature::CategoryManager::Entry, Graphics::GraphicsEntity

- isValid() : WeakPtr< TYPE >
- IsValid() : Base::SkinnedMeshRendererBase, Graphics::View, Graphics::GraphicsEntity, IO::URI, Util::String, Models::ModelNodeInstance, InternalGraphics::InternalGraphicsEntity, Base::ParticleRendererBase, Base::ShaderInstanceBase, Base::VertexLayoutBase, IO::AssignRegistry, IO::BXmlLoaderUtil, Base::MouseRenderDeviceBase, Util::StringAtom, Models::ModelNodeInstance, Debug::DebugCounter, RenderUtil::DrawFullScreenQuad, Util::Blob, Base::VertexLayoutBase, Base::JobPortBase, Base::JobBase, IO::ArchiveBase, Graphics::GraphicsEntity, RenderModules::RTPluginRegistry, Characters::CharacterAnimationController, RenderModules::RenderModule, Base::RenderTargetBase, InternalGraphics::InternalGraphicsEntity, Models::ModelNodeInstance, Graphics::GraphicsEntity, Base::JobPortBase, IO::SchemeRegistry, Resources::ResourceDictionary, Base::ShaderInstanceBase, Base::JobPortBase, Base::JobBase, Characters::CharacterSkeletonInstance, InternalGraphics::InternalGraphicsEntity, Win360::Win360ThreadBarrier, Characters::CharacterInstance, IO::ArchiveBase, Base::SkinnedMeshRendererBase, Util::FourCC, Base::JobPortBase, Base::SkinnedMeshRendererBase, Models::ModelInstance

- isValid() : Ptr< TYPE >
- IsValid() : Graphics::GraphicsEntity, Particles::EnvelopeSampleBuffer, Win32::Win32Guid, Net::DebugMessage

- isValid() : OSX::OSXThreadLocalPtr< TYPE >
IsValid() : Base::ParticleRendererBase, Base::RenderTargetBase, InternalGraphics::InternalGraphicsEntity, Jobs::TPJobThreadPool, FrameSync::FrameSyncTimer, Graphics::GraphicsEntity, Characters::CharacterVariationSet, Base::GameContentServerBase, CoreAnimation::AnimKeyBuffer, Threading::ObjectRef, Animation::AnimSequencer, Particles::EmitterMesh, CoreGraphics::RenderShape, Models::ModelNodeInstance, IO::MediaType, IO::ArchiveBase, Base::VertexLayoutBase
IsValidBool() : Util::String
IsValidFloat() : Util::String
IsValidFloat2() : Util::String
IsValidFloat4() : Util::String
IsValidForRendering() : Characters::CharacterInstance
IsValidHttpRequest() : Http::HttpRequestReader
IsValidHttpResponse() : Http::HttpResponseReader
IsValidInt() : Util::String
IsValidMatrix44() : Util::String
IsVerticalSyncEnabled() : Base::DisplayDeviceBase, Graphics::DisplaySettings
IsVirtual() : BaseGameFeature::CategoryManager::Category
Iterator() : Util::List< TYPE >::Iterator
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Enumerator
- Related Functions

- _
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
- j -

- JobBase() : Base::JobBase
- JobDataDesc() : Jobs::JobDataDesc
- JobPortBase() : Base::JobPortBase
- JobSystem() : Jobs::JobSystem
- JobUniformDesc() : Jobs::JobUniformDesc

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q
- k -

- Key(): Util::KeyValuePair<KEYTYPE, VALUETYPE>
- KeyAtIndex(): Util::Dictionary<KEYTYPE, VALUETYPE>
- KeyboardBase(): Base::KeyboardBase
- KeyCodesByGroup(): Input::Key
- KeyDown(): Base::KeyboardBase
- KeyPressed(): Base::KeyboardBase
- KeysAs(): Util::Dictionary<KEYTYPE, VALUETYPE>
- KeysAsArray(): Util::Dictionary<KEYTYPE, VALUETYPE>
- KeyUp(): Base::KeyboardBase
- KeyValuePair(): Util::KeyValuePair<KEYTYPE, VALUETYPE>
- KeyValuePairAtIndex(): Util::Dictionary<KEYTYPE, VALUETYPE>
- l -

- le() : Math::float2
- Leave() : OSX::OSXCriticalSection, Win360::Win360CriticalSection
- LeaveLockStepMode() :
  FrameSync::FrameSyncHandlerThread, Graphics::GraphicsInterface
- length() : Math::float2, Math::quaternion, Math::line, Math::float4
- Length() : Util::String
- lengthsq() : Math::float2, Math::line, Math::quaternion, Math::float4
- lerp() : Math::float4
- less3_all() : Math::float4
- less3_any() : Math::float4
- less4_all() : Math::float4
- less4_any() : Math::float4
- lessequal3_all() : Math::float4
- lessequal3_any() : Math::float4
- lessequal4_all() : Math::float4
- lessequal4_any() : Math::float4
- Level() : Util::QuadTree< TYPE >::Node
- LevelExists() : BaseGameFeature::CategoryManager
- LightFlickerUtil() : GraphicsFeature::LightFlickerUtil
- LightPrePassServer() : Lighting::LightPrePassServer
- LightServer(): Lighting::LightServer
- LightServerBase(): Lighting::LightServerBase
- Line(): Http::SvgPageWriter
- line(): Math::line
- Line(): Http::SvgPageWriter
- LineBreak(): Http::HtmlPageWriter
- List(): Util::List< TYPE >
- ListDirectories(): IO::IoServer, IO::ZipArchive, OSX::OSXFSWrapper, IO::ArchiveBase, IO::ZipArchive, Win360::Win360FSWrapper
- Listen(): Win360::Win360Socket
- ListFiles(): OSX::OSXFSWrapper, IO::ArchiveBase, IO::ZipArchive, Win360::Win360FSWrapper, IO::IoServer, IO::ZipArchive
- ln(): Math::quaternion
- Load(): Resources::Resource
- load(): Math::matrix44
- Load(): Resources::Resource, Resources::ResourceDictionary, Resources::Resource
- load(): Math::quaternion
- Load(): Resources::Resource, BaseGameFeature::EntityLoader, Resources::Resource, BaseGameFeature::EntityLoaderBase, Resources::Resource, BaseGameFeature::EnvironmentLoader, BaseGameFeature::LevelLoader, Resources::Resource, BaseGameFeature::UserProfile, Resources::Resource
- load(): Math::float4
- Load(): Resources::Resource, Script::Task
- load_float3(): Math::float4
- load_ubyte4n_signed(): Math::float4
- LoadActionFromScript(): Script::ScriptManager
- LoadActionScripts(): Script::ScriptManager
- LoadAssociatedDialogs(): Script::DialogManager
- LoadAttributes(): BaseGameFeature::GlobalAttrsManager
- LoadCanScript(): Script::ScriptManager
- LoadCloseActions(): Script::Task
- LoadCloseConditions(): Script::Task
- LoadCondition(): Script::ScriptManager
- LoadConditionScripts(): Script::ScriptManager
- LoadDialogContent() : Script::Dialog
- LoadDialogs() : Script::DialogManager
- LoadDialogTakeStates() : Script::DialogManager
- LoadEntities() : BaseGameFeature::LoaderServer
- LoaderServer() : BaseGameFeature::LoaderServer
- LoadFailActions() : Script::Task
- LoadFailConditions() : Script::Task
- LoadFailed() : Resources::Resource
- LoadFrameShader() : Frame::FrameShaderLoader
- LoadingResource() : Resources::LoadingResource
- LoadInstances() : BaseGameFeature::CategoryManager
- LoadLevel() : BaseGameFeature::LoaderServer
- LoadManagedModel() : Models::ModelServer
- LoadOnScript() : Script::ScriptManager
- LoadOpenActions() : Script::Task
- LoadOpenConditions() : Script::Task
- LoadResources() : Models::ModelNode, Models::Model, Models::ModelNode, Models::ShapeNode, Models::StateNode, Particles::ParticleSystemNode, Characters::CharacterNode
- LoadState() : Script::ScriptManager
- LoadStateMachine() : Script::ScriptManager
- LoadStateMachines() : Script::ScriptManager
- LoadStatement() : Script::ScriptManager
- LoadStates() : Script::ScriptManager
- LoadSubTasks() : Script::Task
- LoadTableContent() : Script::ScriptManager
- LoadTransition() : Script::ScriptManager
- LoadTransitions() : Script::ScriptManager
- loadu() : Math::float4, Math::matrix44, Math::quaternion, Math::float4
- LocalPath() : IO::URI
- LocalStringAtomTable() : Util::LocalStringAtomTable
- LocalTimeToFileTime() : Base::CalendarTimeBase, Win360::Win360CalendarTime
- Lock() : Resources::Resource
- LockStepModeActive() : FrameSync::FrameSyncHandlerThread
- LockSurfaces() : Resources::D3D9TextureStreamer
- LodDistancesUsed() : Models::TransformNode
- LogFileConsoleHandler() : IO::LogFileConsoleHandler
- LogLevelToString() : Script::InfoLog
- lookatlh() : Math::matrix44
- lookatrh() : Math::matrix44
- LookupAnimatorNodeInstance() : RenderUtil::NodeLookupUtil
- LookupChild() : Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode, Models::ModelNodeInstance, Models::ModelNode
- LookupElements() : Util::SparseTable< TYPE >
- LookupFrameShader() : Frame::FrameServer
- LookupManagedModel() : Models::ModelServer
- LookupManagedResource() : Resources::ResourceManager
- LookupNode() : Models::Model
- LookupNodeInstance() : Models::ModelInstance
- LookupPath() : Models::ModelNodeInstance
- LookupProjectDirectory() : App::RenderApplication
- LookupResource() : Resources::ResourceManager
- LookupSamples() : Particles::EnvelopeSampleBuffer
- LookupStateNodeInstance() : RenderUtil::NodeLookupUtil
- lt() : Math::float2
- m -

- ManagedResource(): Resources::ManagedResource
- Manager(): Game::Manager
- Map(): IO::MemoryStream, Base::IndexBufferBase, Win360::D3D9IndexBuffer, Base::TextureBase, Direct3D9::D3D9Texture, Base::TextureBase, Win360::D3D9VertexBuffer, Base::TextureBase, Base::VertexBufferBase, Direct3D9::D3D9Texture, IO::FileStream, IO::ZipFileStream, Win360::D3D9IndexBuffer, Win360::D3D9VertexBuffer, IO::Stream, CoreAnimation::AnimKeyBuffer
- MapCubeFace(): Base::TextureBase, Direct3D9::D3D9Texture, Base::TextureBase, Direct3D9::D3D9Texture
- MapInfo(): Base::TextureBase::MapInfo
- MapSampleCounts(): CoreAnimation::AnimSampleBuffer
- MapSamples(): CoreAnimation::AnimSampleBuffer
- MarkRemove(): InternalGraphics::InternalGraphicsEntity
- MatchPattern(): Util::String
- matrix44(): Math::matrix44
- maximize(): Math::float4, Math::float2
- MayaCameraProperty(): GraphicsFeature::MayaCameraProperty
- MayaCameraUtil(): RenderUtil::MayaCameraUtil
- MediaType(): IO::MediaType
- `MemoryIndexBufferLoaderBase()`: \texttt{Base::MemoryIndexBufferLoaderBase}
- `MemoryMeshLoader()`: \texttt{CoreGraphics::MemoryMeshLoader}
- `MemoryPageHandler()`: \texttt{Debug::MemoryPageHandler}
- `MemoryStream()`: \texttt{IO::MemoryStream}
- `MemoryVertexBufferLoaderBase()`: \texttt{Base::MemoryVertexBufferLoaderBase}
- `MeshBase()`: \texttt{Base::MeshBase}
- `MeshPageHandler()`: \texttt{Debug::MeshPageHandler}
- `Message()`: \texttt{Messaging::Message}
- `MessageBox()`: \texttt{Win32::SysFunc, OSX::SysFunc}
- `MessageClient()`: \texttt{Net::MessageClient}
- `MessageReader()`: \texttt{Messaging::MessageReader}
- `MessageWriter()`: \texttt{Messaging::MessageWriter}
- `minimize()`: \texttt{Math::float4, Math::float2}
- `Model()`: \texttt{Models::Model}
- `ModelEntity()`: \texttt{Graphics::ModelEntity}
- `ModelEntityShared()`: \texttt{Shared::ModelEntityShared}
- `ModelInstance()`: \texttt{Models::ModelInstance}
- `ModelNode()`: \texttt{Models::ModelNode}
- `ModelNodeInstance()`: \texttt{Models::ModelNodeInstance}
- `ModelServer()`: \texttt{Models::ModelServer}
- `ModulateStepTime()`: \texttt{Particles::ParticleSystem}
- `MonthToString()`: \texttt{Base::CalendarTimeBase}
- `MountArchive()`: \texttt{IO::IoServer}
- `MountStandardArchives()`: \texttt{IO::IoServer}
- `MouseBase()`: \texttt{Base::MouseBase}
- `MouseGripperProperty()`: \texttt{PhysicsFeature::MouseGripperProperty}
- `MousePointer()`: \texttt{CoreGraphics::MousePointer}
- `MouseRenderDevice()`: \texttt{CoreGraphics::MouseRenderDevice}
- `MouseRenderDeviceBase()`: \texttt{Base::MouseRenderDeviceBase}
- `MouseRenderer()`: \texttt{Graphics::MouseRenderer}
- `MultipleRenderTargetBase()`: \texttt{Base::MultipleRenderTargetBase}
- `multiply()`: \texttt{Math::matrix44, Math::float4, Math::quaternion}`
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Enumerator
- Related Functions

- _
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
NavigateToNextLayer() : Script::Dialog
nearequal3() : Math::float4
nearequal4() : Math::float4
NeedsLightModelLinking() : Lighting::LightServerBase, Lighting::LightPrePassServer, Lighting::LightServerBase
Next() : Core::ExitHandler
NextString() : Util::StringBuffer
NextToken() : Script::ActionReader
Node() : Util::SimpleTree<VALUETYPE>::Node, Util::QuadTree<TYPE>::Node, Util::SimpleTree<VALUETYPE>::Node
normalize() : Math::float2, Math::float4, Math::plane, Math::float4, Math::quaternion
NoSlotFound() : Resources::PoolResourceMapper
Notify() : FSM::StateMachine, FSM::Transition, Actions::Action, Conditions::FSMCondition, Actions::Action, Actions::FSMAction, Actions::SequenceAction, FSM::State
NotifyEventHandlers() : Base::RenderDeviceBase, Base::DisplayDeviceBase, Base::RenderDeviceBase, Base::DisplayDeviceBase, Base::RenderDeviceBase
NotifyGameLoad() : Game::GameServer
NotifyGameSave() : Game::GameServer
NotifyOfEntityTransformChange() :
InternalGraphics::InternalStage
- nullvec() : Math::vector
- NumberOfPrimitives() : CoreGraphics::PrimitiveTopology
- NumberOfVertices() : CoreGraphics::PrimitiveTopology
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q
ObjectInspectorHandler() : Debug::ObjectInspectorHandler
ObjectRef() : Threading::ObjectRef
ObserverContext() : Visibility::ObserverContext
ObtainKeyboardCapture() : Base::InputServerBase
ObtainMouseCapture() : Base::InputServerBase
OnActivate() : Game::Manager, Game::Property,
BaseGameFeature::TimeManager,
BaseGameFeature::CategoryManager,
BaseGameFeature::TimeSource,
BaseGameFeature::EnvQueryManager,
BaseGameFeature::GlobalAttrsManager, Game::Entity,
InternalGraphics::InternalGraphicsEntity, Game::FeatureUnit,
InternalGraphics::InternalGraphicsEntity, Game::Manager,
Game::Property, InternalGraphics::InternalGraphicsEntity,
GraphicsFeature::ChaseCameraProperty,
GraphicsFeature::GraphicsProperty,
GraphicsFeature::MayaCameraProperty, Game::Property,
PhysicsFeature::ActorPhysicsProperty,
PhysicsFeature::MouseGripperProperty, Game::Property,
PhysicsFeature::PhysicsProperty,
PhysicsFeature::TriggerProperty, Actions::Action,
Actions::ActionList, Actions::FSMAction,
Actions::SequenceAction, Conditions::FSMCondition,
Game::Manager, FSM::Transition, Script::DialogManager,
Script::ScriptManager, ScriptFeature::ScriptFeatureUnit, Game::Property, StateObjectFeature::StateGraphicsProperty, Actions::Action, Game::Manager, InternalGraphics::InternalGraphicsEntity, InternalGraphics::InternalModelEntity, BaseGameFeature::TimeSource, GraphicsFeature::GraphicsProperty

- **OnAttach()**: XInput::XInputGamePad, Base::KeyboardBase, Base::MouseBase, CoreGraphics::DisplayEventHandler, CoreGraphics::RenderEventHandler, Base::GamePadBase, Base::KeyboardBase, Base::MouseBase, XInput::XInputGamePad, Input::InputHandler, CoreGraphics::RenderEventHandler
- **OnAttachedToSequencer()**: Animation::AnimJob
- **OnAttachEntity()**: RenderModules::RTPlugin, RenderModules::RTPluginRegistry
- **OnAttachToModel()**: Characters::CharacterNode, Models::ModelNode, Models::TransformNode, Particles::ParticleSystemNode, Models::ModelNode
- **OnAttachToResource()**: Resources::ResourceSaver, Resources::ResourceLoader, Resources::ResourceSaver, Resources::ResourceLoader
- **OnAttachToResourceManager()**: Resources::ResourceMapper, Resources::SimpleResourceMapper, Resources::PoolResourceMapper
- **OnAttachToServer()**: InternalGraphics::InternalStage, InternalGraphics::InternalView
- **OnAttachToStage()**: InternalGraphics::InternalGraphicsEntity
- **OnAttachToView()**: Graphics::CameraEntity, InternalGraphics::InternalCameraEntity
- **OnBeginFrame()**: Game::Property, Game::Manager, Game::Property, Input::InputHandler, XInput::XInputGamePad, Base::KeyboardBase, Base::MouseBase, Base::SkinnedMeshRendererBase, Base::MouseBase, BaseGameFeature::EntityManager, Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property, PhysicsFeature::MouseGripperProperty, Game::Property, Game::FeatureUnit, Game::Manager, PhysicsFeature::TriggerProperty, Game::Manager,
Game::FeatureUnit, Game::Property,
StateObjectFeature::StateProperty, Game::Manager,
Game::FeatureUnit, Game::Manager, Base::KeyboardBase,
Game::Manager, Base::MouseBase, Game::Manager,
Input::InputHandler, XInput::XInputGamePad,
Base::SkinnedMeshRendererBase, Game::Property
- OnCancelRequest() : Resources::LoadingResource
- OnChar() : Win32::Win32DisplayDevice
- OnCloseRequested() : Win32::Win32DisplayDevice
- OnConfigureDisplay() : App::RenderApplication
- OnCopyFile() : IO::IoInterfaceHandler
- OnCreateDirectory() : IO::IoInterfaceHandler
- OnCreateManagedResource() : Resources::ResourceMapper,
  Resources::SimpleResourceMapper
- OnCullBefore() : InternalGraphics::InternalGraphicsEntity,
  InternalGraphics::InternalModelEntity,
  InternalGraphics::InternalStage
- OnDeactivate() : Game::Manager,
  GraphicsFeature::CameraProperty,
  BaseGameFeature::TimeManager,
  BaseGameFeature::TimeSource,
  BaseGameFeature::CategoryManager,
  BaseGameFeature::EntityManager,
  BaseGameFeature::EnvEntityManager,
  BaseGameFeature::EnvQueryManager, Game::Entity,
  InternalGraphics::InternalGraphicsEntity, Game::FeatureUnit,
  InternalGraphics::InternalGraphicsEntity, Game::Manager,
  InternalGraphics::InternalGraphicsEntity, Game::Property,
  GraphicsFeature::CameraProperty,
  GraphicsFeature::GraphicsProperty, Game::Property,
  GraphicsFeature::MayaCameraProperty,
  PhysicsFeature::ActorPhysicsProperty,
  PhysicsFeature::EnvironmentCollideProperty,
  PhysicsFeature::MouseGripperProperty,
  PhysicsFeature::PhysicsProperty,
  PhysicsFeature::TriggerProperty, Actions::Action,
  Actions::ActionList, Actions::FSMAction,
  Actions::SequenceAction, Conditions::FSMCondition,
  Game::Manager, FSM::Transition, Script::DialogManager,
Script::ScriptManager, ScriptFeature::ScriptFeatureUnit, StateObjectFeature::StateGraphicsProperty, StateObjectFeature::StateProperty, Actions::Action, Game::Manager, BaseGameFeature::TimeSource, Game::Manager, BaseGameFeature::TimeSource, Game::Property, InternalGraphics::InternalCameraEntity, InternalGraphics::InternalGraphicsEntity, InternalGraphics::InternalModelEntity, GraphicsFeature::GraphicsProperty

- OnDeleteDirectory() : IO::IoInterfaceHandler
- OnDeleteFile() : IO::IoInterfaceHandler
- OnDiscardManagedResource() : Resources::ResourceMapper, Resources::SimpleResourceMapper
- OnDiscardSharedData() :
- OnDiscardStage() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry
- OnDiscardView() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry
- OnEndFrame() : Game::Manager, Input::InputHandler, BaseGameFeature::EntityManager, Input::InputHandler, Game::FeatureUnit, Base::SkinnedMeshRendererBase, Game::Manager, Game::FeatureUnit, Game::Manager, Game::FeatureUnit, Game::Manager, Input::InputHandler, Game::Manager, Base::SkinnedMeshRendererBase
- OnEntry() : FSM::State
- OnEvent() : Input::InputHandler, Base::KeyboardBase, Base::MouseBase, Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- OnExit() : FSM::State, Core::ExitHandler
- OnFrame() : BaseGameFeature::TimeManager, Win32::Win32InputServer, BaseGameFeature::EnvQueryManager, BaseGameFeature::FocusManager, BaseGameFeature::GameManager,
Animation::AnimEventHandlerBase,
Animation::AnimEventServer, Game::FeatureUnit,
Game::GameServer, Game::Manager,
GraphicsFeature::AttachmentManager, Game::Manager,
FSM::State, Game::FeatureUnit, Debug::DebugTextRenderer
, Game::Manager, Debug::DebugShapeRenderer,
Game::Manager, Graphics::GraphicsServer,
Base::InputServerBase, Game::Manager, Http::HttpServer,
Win32::Win32InputServer,
InternalGraphics::AttachmentServer,
InternalGraphics::InternalGraphicsServer,
RenderModules::RenderModule

- OnGainActivity() : Game::Property, Game::Entity,
  Game::Property
- OnHide() : InternalGraphics::InternalGraphicsEntity,
  Models::ModelNodeInstance,
  InternalGraphics::InternalGraphicsEntity,
  Models::ModelNodeInstance,
  InternalGraphics::InternalModelEntity,
  Models::ModelNodeInstance, Models::ModelInstance,
  Particles::ParticleSystemNodeInstance
- OnKeyDown() : Win32::Win32DisplayDevice
- OnKeyUp() : Win32::Win32DisplayDevice
- OnKillFocus() : Win32::Win32DisplayDevice
- OnLoad() : Game::Manager, Game::Property,
  Game::FeatureUnit, Game::Property,
  BaseGameFeature::TimeManager,
  BaseGameFeature::TimeSource,
  BaseGameFeature::EntityManager, Game::Entity,
  Game::FeatureUnit, Game::Manager, Game::Property,
  Game::Manager, Game::Property,
  PhysicsFeature::TriggerProperty, Game::Manager,
  Game::Property, Script::DialogManager,
  StateObjectFeature::StateProperty, Game::FeatureUnit,
  Game::Manager, Game::FeatureUnit, Game::Manager,
  Actions::SequenceAction, Game::Manager,
  BaseGameFeature::TimeSource, Game::Manager,
  BaseGameFeature::TimeSource, Game::Property
- OnLoadCancelled() : Resources::ResourceLoader,
Resources::StreamResourceLoader,
Resources::ResourceLoader,
Resources::StreamResourceLoader,
Models::StreamModelLoader

- OnLoadRequested() : Resources::D3D9TextureStreamer,
  Resources::ResourceLoader,
  Resources::StreamResourceLoader,
  Resources::D3D9TextureStreamer,
  Win360::D3D9MemoryIndexBufferLoader,
  Win360::D3D9MemoryVertexBufferLoader,
  Resources::StreamResourceLoader,
  CoreGraphics::MemoryMeshLoader,
  Resources::ResourceLoader,
  Win360::D3D9MemoryIndexBufferLoader,
  Resources::ResourceLoader,
  Win360::D3D9MemoryVertexBufferLoader,
  Resources::StreamResourceLoader,
  Models::StreamModelLoader

- OnLoadResources() :
  StateObjectFeature::StateGraphicsProperty

- OnLoseActivity() : Game::Property, Game::Entity,
  Game::Property, PhysicsFeature::ActorPhysicsProperty, Game::Property

- OnLoseCameraFocus() : GraphicsFeature::CameraProperty

- OnLoseInputFocus() : GraphicsFeature::MayaCameraProperty

- OnLostDevice() : Direct3D9::D3D9ShaderInstance

- OnMinimized() : Win32::Win32DisplayDevice

- OnMountArchive() : IO::IoInterfaceHandler

- OnMouseButton() : Win32::Win32DisplayDevice

- OnMouseMove() : Win32::Win32DisplayDevice

- OnMouseWheel() : Win32::Win32DisplayDevice

- OnMoveAfter() : Game::Property, Game::Entity,
  PhysicsFeature::MouseGripperProperty, Game::Property,
  PhysicsFeature::ActorPhysicsProperty, Game::Property,
  PhysicsFeature::PhysicsProperty, Game::Property

- OnMoveBefore() : Game::Property, Game::Entity,
  Game::Property, PhysicsFeature::ActorPhysicsProperty,
  PhysicsFeature::MouseGripperProperty, Game::Property

- OnNotifyCullingVisible() :

- OnObtainCameraFocus() : GraphicsFeature::CameraProperty, GraphicsFeature::ChaseCameraProperty
- OnObtainCapture() : Input::InputHandler, Base::KeyboardBase, Base::MouseBase, Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- OnObtainInputFocus() : GraphicsFeature::MayaCameraProperty
- OnPaint() : Win32::Win32DisplayDevice
- OnPending() : Resources::ResourceLoader, Resources::StreamResourceLoader, Resources::ResourceLoader, Resources::StreamResourceLoader, Resources::ResourceLoader, Resources::StreamResourceLoader, Models::StreamModelLoader
- OnPrepare() : Resources::ResourceMapper, Resources::SimpleResourceMapper, Resources::PoolResourceMapper
- OnProcessInput() : App::RenderApplication, App::ViewerApplication
- OnReadStream() : IO::IoInterfaceHandler
- OnRegister() : RenderModules::RTPlugin
- OnReleaseCapture() : Input::InputHandler, Base::KeyboardBase, Base::MouseBase, Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- OnRemove() : Input::InputHandler, CoreGraphics::DisplayEventHandler, Input::InputHandler, CoreGraphics::DisplayEventHandler, Input::InputHandler,
CoreGraphics::RenderEventHandler
- OnRemoveEntity() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry
- OnRemoveFromMapper() : Resources::ResourceScheduler, Resources::TexturePoolMapperScheduler
- OnRemoveFromModel() : Models::ModelNode, Particles::ParticleSystemNode, Characters::CharacterNode
- OnRemoveFromResource() : Resources::ResourceLoader, Resources::ResourceSaver, Resources::ResourceLoader, Resources::ResourceSaver
- OnRemoveFromResourceManager() : Resources::ResourceMapper, Resources::SimpleResourceMapper, Resources::PoolResourceMapper
- OnRemoveFromSequencer() : Animation::AnimJob
- OnRemoveFromServer() : InternalGraphics::InternalStage, InternalGraphics::InternalView
- OnRemoveFromStage() : InternalGraphics::InternalGraphicsEntity
- OnRemoveFromView() : Graphics::CameraEntity, InternalGraphics::InternalCameraEntity
- OnRender() : Game::Property, GraphicsFeature::CameraProperty, Game::Property, Game::Entity, Game::Property, GraphicsFeature::CameraProperty, Game::Property, GraphicsFeature::MayaCameraProperty, Game::Property, GraphicsFeature::ChaseCameraProperty, Game::Property
- OnRenderAfter() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry
OnRenderDebug() : Game::Manager, Game::Property, Game::FeatureUnit, Game::Property, Visibility::VisibilityChecker, GraphicsFeature::GraphicsProperty, Visibility::VisibilityClusterSystem, Visibility::VisibilityBoxSystem, Visibility::VisibilityQuadtree, Visibility::VisibilitySystemBase, BaseGameFeature::TimeManager, Game::Property, Game::Entity, Game::Property, BaseGameFeature::EntityManager, InternalGraphics::InternalGraphicsEntity, Game::Property, PhysicsFeature::TriggerProperty, Lighting::InternalPointLightEntity, Game::Manager, Game::Property, Game::Manager, Game::Property, InternalGraphics::InternalGraphicsEntity, Game::Manager, Game::FeatureUnit, Game::Manager, Game::FeatureUnit, Game::Property, Game::Manager, GraphicsFeature::GraphicsProperty, Game::Property, Game::Manager, InternalGraphics::InternalModelEntity, PhysicsFeature::ActorPhysicsProperty, InternalGraphics::InternalStage, Lighting::InternalAbstractLightEntity, PhysicsFeature::MouseGripperProperty, Game::Property, Lighting::InternalSpotLightEntity, Lighting::InternalGlobalLightEntity, GraphicsFeature::GraphicsProperty, Game::Manager

OnRenderFrameBatch() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry

OnRenderWithoutView() : RenderModules::RTPluginRegistry, RenderModules::RTPlugin

OnRequestManagedResource() : Resources::TexturePoolMapperScheduler, Resources::ResourceScheduler

OnRequestOtherMipMap() : Resources::TexturePoolMapperScheduler

OnReset() : Base::GamePadBase, Base::KeyboardBase, Base::GamePadBase, Base::KeyboardBase, Base::MouseBase, Input::InputHandler, Base::MouseBase

OnResetDevice() : Direct3D9::D3D9ShaderInstance
- OnResetSharedData() :
  - InternalGraphics::InternalGraphicsEntity
  - InternalGraphics::InternalModelEntity
- OnResolveVisibility() :
  - InternalGraphics::InternalGraphicsEntity
  - Lighting::InternalAbstractLightEntity
  - InternalGraphics::InternalModelEntity
  - Lighting::InternalAbstractLightEntity
- OnResourcesLoaded() :
  - Models::ModelNode
  - Characters::CharacterNode
  - Models::ModelNode
  - Particles::ParticleSystemNode
- OnRestored() :
  - Win32::Win32DisplayDevice
- OnSave() :
  - Game::Manager
  - Game::Property
  - Game::FeatureUnit
  - Game::Property
  - BaseGameFeature::TimeManager
  - StateObjectFeature::StateProperty
  - Resources::ResourceSaver
  - Game::Property
  - Game::Entity
  - Game::FeatureUnit
  - BaseGameFeature::TimeSource
  - Game::Manager
  - Game::Property
  - Base::StreamTextureSaverBase
  - Game::Property
  - Game::Manager
  - Game::Property
  - Win360::D3D9StreamTextureSaver
  - BaseGameFeature::EntityManager
  - Game::Manager
  - Game::Property
  - Game::Manager
  - Game::FeatureUnit
  - Win360::D3D9StreamTextureSaver
  - Game::Property
  - Game::Manager
  - Game::FeatureUnit
  - Game::Manager
  - PhysicsFeature::TriggerProperty
  - Game::Manager
  - BaseGameFeature::TimeSource
  - Game::Manager
  - Game::Property
  - BaseGameFeature::TimeSource
  - Game::Property
  - Game::Property
  - Script::DialogManager
  - Game::FeatureUnit
  - Game::Property
- OnSetCursor() :
  - Win32::Win32DisplayDevice
- OnSetFocus() :
  - Win32::Win32DisplayDevice
- OnSetOverwriteColor() :
  - GraphicsFeature::GraphicsProperty
- OnSetShaderVariable() :
  - GraphicsFeature::GraphicsProperty
- OnSetupResourceMappers() :
  - App::RenderApplication
- OnSetupSharedData() :
  - InternalGraphics::InternalGraphicsEntity
  - Graphics::GraphicsEntity
InternalGraphics::InternalModelEntity, Graphics::GraphicsEntity
- OnStageCreated() : RenderModules::RTPlugin, RenderModules::RTPluginRegistry
- OnStart() : GraphicsFeature::CameraProperty, Game::Property, Game::Manager, Game::Property, Game::FeatureUnit, Game::Manager, GraphicsFeature::CameraProperty, Game::Property, Game::FeatureUnit, Game::Manager, Game::Property, GraphicsFeature::CameraProperty, Game::Entity, GraphicsFeature::CameraProperty, Game::FeatureUnit, Game::Property, Game::Manager, Game::Property, Game::Manager, Game::Property, Game::Manager, Game::Property, BaseGameFeature::EntityManager, Game::Property
- OnStateEnter() : BaseGameFeature::GameStateHandler
- OnStateLeave() : BaseGameFeature::GameStateHandler
- OnSuccessRequest() : Resources::LoadingResource, Resources::PoolLoadingResource
- OnSwitchActiveState() : StateObjectFeature::StateGraphicsProperty
- OnToggleFullscreenWindowed() : Win32::Win32DisplayDevice
- OnUnregister() : `RenderModules::RTPlugin`
- OnUpdate() : `Resources::SimpleResourceMapper`, `Resources::ResourceMapper`
- OnUpdateAfter() : `RenderModules::RTPluginRegistry`, `RenderModules::RTPlugin`
- OnUpdateBefore() : `RenderModules::RTPluginRegistry`, `RenderModules::RTPlugin`
- OnUpdateFrame() : `App::RenderApplication`, `App::ViewerApplication`
- OnViewCreated() : `RenderModules::RTPlugin`, `RenderModules::RTPluginRegistry`
- OnVisibilityResolve() : `Models::ModelNodeInstance`, `Models::ShapeNodeInstance`, `Models::ModellInstance`, `Models::ShapeNodeInstance`, `Particles::ParticleSystemNodeInstance`
- OnWriteStream() : `IO::IoInterfaceHandler`
- Open() : `App::ConsoleApplication`, `IO::Stream`, `IO::FileStream`, `Win32::Win32DisplayDevice`, `Base::VertexLayoutServerBase`, `Game::GameServer`, `Win360::Win360Socket`, `IO::ConsoleHandler`, `IO::StreamReader`, `Lighting::LightPrePassServer`, `Debug::DebugHandler`, `Direct3D9::D3D9RenderDevice`, `BaseGameFeature::LoaderServer`, `FSM::StateMachine`, `Http::HtmlPageWriter`, `Visibility::VisibilitySystemBase`, `Visibility::VisibilityBoxSystem`, `Animation::AnimEventServer`, `Lighting::SM30ShadowServer`, `IO::BXmlReader`, `Http::SvgLineChartWriter`, `IO::ConsoleHandler`, `Win32::Win32InputServer`, `IO::Console`, `Graphics::Display`, `Messaging::Handler`, `Base::DisplayDeviceBase`, `Visibility::VisibilityChecker`, `Messaging::AsyncPort`, `Base::ShapeRendererBase`, `Lighting::LightPrePassServer`, `Base::TransformDeviceBase`, `IO::LogFileConsoleHandler`, `Direct3D9::D3D9TextRenderer`, `IO::StreamReader`, `Direct3D9::D3D9TextRenderer`, `Base::RenderDeviceBase`, `App::RenderApplication`, `IO::BinaryReader`, `IO::ZipFileStream`, `IO::StreamWriter`, `Visibility::VisibilityQuadtree`, `IO::StreamWriter`, `IO::IoInterfaceHandler`, `Http::SvgPageWriter`, `Http::HttpServer`, `Lighting::SM30LightServer`, `...`
IO::ConsoleHandler, Win360::D3D9ShapeRenderer, Win360::D3D9TransformDevice, Messaging::Handler, App::ViewerApplication, Debug::DebugInterface, Frame::FrameServer, Graphics::GraphicsHandler, Graphics::GraphicsInterface, Graphics::GraphicsServer, IO::BinaryWriter, App::Application, IO::ZipFileEntry, IO::StreamWriter, Net::StdTcpServer, App::GameApplication, Win32::Win32InputServer, Messaging::Handler, Direct3D9::D3D9ShaderServer, InternalGraphics::InternalGraphicsServer, Interface::InterfaceBase, Net::StdTcpServer, Base::ShaderServerBase, IO::StreamReader, IO::MemoryStream, Base::VertexLayoutServerBase, IO::ExcelXmlReader, IO::XmlReader, Resources::ResourceManager, Lighting::SM30ShadowServer, Win32::Win32DisplayDevice, Lighting::LightServerBase, Base::InputServerBase, Core::CoreServer, InternalGraphics::AttachmentServer, Http::HttpMessageHandler, Win360::Win360Socket, IO::ConsoleHandler, Direct3D9::D3D9ShaderServer, Http::HttpInterface, Win32::Win32DisplayDevice, Messaging::Handler, Models::ModelServer, IO::StreamReader, Visibility::VisibilityClusterSystem, Particles::ParticleServer, Win360::D3D9TransformDevice, IO::ConsoleHandler, IO::StreamWriter, Base::TextRendererBase, IO::XmlWriter, Http::HttpServerProxy, Win360::D3D9ShapeRenderer, Models::VisResolver, IO::StreamWriter
- OpenDInputMouse() : Win32::Win32InputServer
- OpenFile() : Win360::Win360FSWrapper, OSX::OSXFSWrapper
- OpenProgressIndicator() : BaseGameFeature::LoaderServer
- OpenWindow() : Win32::Win32DisplayDevice
- Operator() : Conditions::Operator< TYPE >
- operator *() : Util::List< TYPE >::Iterator, Math::float4, WeakPtr< TYPE >, Ptr< TYPE >, Math::float4, Math::vector, Math::float2
- operator *=( ) : Math::vector, Math::float4, Math::float2, Math::float4
- operator bool() : Util::List<TYPE>::Iterator
- operator delete() : Util::String, Win32::Win32Guid, OSX::OSXGuid, Util::Blob
- operator new() : Util::Blob, OSX::OSXGuid, Win32::Win32Guid, Util::String
- operator TYPE *() : WeakPtr<TYPE>, Ptr<TYPE>
- operator!=() : Math::matrix44, Util::FixedArray<TYPE>, Math::float4, Util::Variant, Core::Rtti, Util::Blob, IO::URI, Util::Array<TYPE>, IO::MediaType, Util::StringAtomTableBase::StaticString, Ptr<TYPE>, Util::Variant, Util::StringAtom, Util::Queue<TYPE>, Win32::Win32Guid, Util::Variant, Math::vector, Util::StringAtom, Util::Variant, Util::FourCC, Util::Variant, CoreGraphics::DisplayMode, Util::FixedTable<TYPE>, Math::float4, Util::BitField<NUMBITS>, Ptr<TYPE>, Math::quaternion, Util::KeyValuePair<KEYTYPE, VALUETYPE>, Util::Variant, Util::Queue<TYPE>, Math::point, Util::Stack<TYPE>, Math::float2, Util::List<TYPE>::Iterator, Util::Array<TYPE>, OSX::OSXGuid, Util::StringAtom, Math::float4
- operator()() : Util::Delegate<ARGTYPE>
- operator+() : Math::float4, Math::vector, Math::float4, Math::float2, Math::point
- operator++() : Util::List<TYPE>::Iterator
- operator+=() : Math::point, Math::float4, Math::float2, Math::vector, Math::float4, Util::String, Math::float4
- operator()-() : Math::float4, Math::point, Math::float4, Math::vector, Math::float2, Math::vector, Math::float4, Math::float2, Math::point
- operator--() : Util::List<TYPE>::Iterator
- operator-() : Math::float4, Math::vector, Math::float2, Math::point
- operator->() : WeakPtr<TYPE>, Util::List<TYPE>::Iterator, Ptr<TYPE>
- operator() : Util::StringAtom, Util::Variant, Util::KeyValuePair<KEYTYPE, VALUETYPE>, Util::Blob, OSX::OSXGuid, Util::FourCC, Util::StringAtomTableBase::StaticString, Win360::Win360IpAddress, Win32::Win32Guid
- operator\(\leq()\) : Win32::Win32Guid, Util::FourCC, OSX::OSXGuid, Util::Variant, Util::KeyValuePair<KEYTYPE, VALUETYPE>, Util::StringAtom
- operator\(\geq()\) : Util::FixedTable<ETYPE>, Ptr<ETYPE>, Util::Variant, Util::Dictionary<KEYTYPE, VALUETYPE>, Math::quaternion, Util::BitField<NUMBITS>, Ptr<ETYPE>, Util::Variant, Util::List<ETYPE>, WeakPtr<ETYPE>, Math::vector, Util::Variant, Math::point, Util::RingBuffer<ETYPE>, Util::Variant, Win32::Win32Guid, Util::Variant, Math::vector, Util::String, Math::float4, Util::StringAtom, Util::FixedArray<ETYPE>, Util::StringAtom, IO::URI, Math::matrix44, Util::Variant, Util::KeyValuePair<KEYTYPE, VALUETYPE>, Util::Variant, Threading::SafePriorityQueue<PRITYPE, TYPE>, Util::Array<ETYPE>, Util::Variant, Util::String, Math::quaternion, Threading::SafeQueue<ETYPE>, IO::MediaType, WeakPtr<ETYPE>, Util::Stack<ETYPE>, Util::Blob, Util::Variant, Util::List<ETYPE>:Iterator, Util::Queue<ETYPE>, Util::Variant, Win32::Win32Guid, Math::point, Util::Variant, WeakPtr<ETYPE>, Util::Variant, Math::matrix44, OSX::OSXGuid, Util::Variant, Math::polar, Util::HashTable<KEYTYPE, VALUETYPE>, Math::float4, Util::Variant
- operator\(\!=()\) : Core::Rtti, Win360::Win360IpAddress, Math::float4, Util::FixedTable<ETYPE>, Math::vector, Util::Variant, Util::List<ETYPE>:Iterator, Util::Variant, Util::Queue<ETYPE>, Util::Stack<ETYPE>, Util::Variant, OSX::OSXGuid, Util::Array<ETYPE>, Util::Queue<ETYPE>, Util::KeyValuePair<KEYTYPE, VALUETYPE>, Util::quaternion, Math::point, Math::float4, Util::StringAtom, CoreGraphics::DisplayMode, Win32::Win32Guid, Util::Variant, Util::Blob, IO::MediaType, Util::Variant, Ptr<ETYPE>, Util::BitField<NUMBITS>, Util::StringAtomTableBase::StaticString, Math::float4, Ptr<ETYPE>, Util::Variant, Util::FourCC, Math::matrix44, Math::float2, Util::Array<ETYPE>, Util::StringAtom, Util::FixedSizeArray<ETYPE>, Messaging::Id, Util::Variant, Util::StringAtom, IO::URI
- operator\(>()\) : Util::StringAtom, Util::FourCC, Util::Variant,
Win32::Win32Guid, Util::StringAtomTableBase::StaticString, Util::Blob, Win360::Win360IpAddress, Util::KeyValuePair<KEYTYPE, VALUETYPE>, OSX::OSXGuid

- operator>=() : Util::StringAtom, Win32::Win32Guid, Util::FourCC, Util::Blob, Util::Variant, OSX::OSXGuid, Util::KeyValuePair<KEYTYPE, VALUETYPE>
- operator[]( ) : Util::FixedArray<TYPE>, Util::HashTable<KEYTYPE, VALUETYPE>, Util::Dictionary<KEYTYPE, VALUETYPE>, Util::RingBuffer<TYPE>, Util::String, Util::Dictionary<KEYTYPE, VALUETYPE>, Util::Queue<TYPE>, Util::SimpleTree<VALUETYPE>::Node, Util::Stack<TYPE>, Util::Array<TYPE>, Util::SimpleTree<VALUETYPE>::Node, Util::Queue<TYPE>, Util::Array<TYPE>

- Or() : Util::BitField<NUMBITS>
- origin() : Math::point
- ortholh() : Math::matrix44
- orthooffcenterlh() : Math::matrix44
- orthooffcenterrh() : Math::matrix44
- orthorh() : Math::matrix44
- OSXCriticalSection() : OSX::OSXCriticalSection
- OSXFileTime() : OSX::OSXFileTime
- OSXGuid() : OSX::OSXGuid
- OSXHeap() : OSX::OSXHeap
- OSXMemoryPool() : OSX::OSXMemoryPool
- OSXThread() : OSX::OSXThread
- OSXThreadLocalPtr() : OSX::OSXThreadLocalPtr<TYPE>
- OverwriteScriptManager() : ScriptFeature::ScriptFeatureUnit
- Owner() : FrameSync::FrameSyncSharedData
- OwnerDiscard() : FrameSync::FrameSyncSharedData
- OwnerSetup() : FrameSync::FrameSyncSharedData
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
typedefs
Enumerations
Enum
Related Functions

_  
 a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q
- p -

- Parent() : Util::SimpleTree< VALUETYPE >::Node
- ParseArgs() : Actions::Action, Conditions::Condition, Actions::Action, Conditions::Condition
- ParseBluePrints() : BaseGameFeature::FactoryManager
- ParseDataTag() : Characters::CharacterNode, Characters::CharacterSkinNode, Models::ShapeNode, Models::StateNode, Models::TransformNode, Particles::ParticleSystemNode, Models::ModelNode
- ParseFile() : StateObjectFeature::StateProperty
- ParseKeyValuePairs() : Util::String
- ParseQuery() : IO::URI
- ParticleRenderer() : Particles::ParticleRenderer
- ParticleRendererBase() : Base::ParticleRendererBase
- ParticleServer() : Particles::ParticleServer
- ParticleSystem() : Particles::ParticleSystem
- ParticleSystemNode() : Particles::ParticleSystemNode
- ParticleSystemNodeInstance() : Particles::ParticleSystemNodeInstance
- PatchInputDesc() : Base::JobBase
- PatchOutputDesc() : Base::JobBase
- PatchUniformDesc() : Base::JobBase
- Pause() : BaseGameFeature::TimeSource, Debug::DebugTimer, BaseGameFeature::TimeSource
- PauseAll() : BaseGameFeature::TimeManager
- Peek() : Win360::Win360Event, Messaging::AsyncPort, Util::Queue<TYPE>, Threading::SafePriorityQueue<PRITYPE, TYPE>, Threading::SafeQueue<TYPE>, Messaging::AsyncPort, Util::Stack<TYPE>, Messaging::AsyncPort, Win360::Win360Event
- PerformVisibilityQuery() : Visibility::VisibilityChecker
- permute() : Math::float4
- permute_control() : Math::float4
- perspfovlh() : Math::matrix44
- perspfovrh() : Math::matrix44
- persph() : Math::matrix44
- perspoffcenterlh() : Math::matrix44
- perspoffcenterrh() : Math::matrix44
- persprh() : Math::matrix44
- PhysicsProperty() : PhysicsFeature::PhysicsProperty
- plane() : Math::plane
- PlatformAsString() : Base::SystemInfoBase
- PlayClipJob() : Animation::PlayClipJob
- PlayEmotes() : Script::DialogManager
- PlaySound() : Script::DialogManager
- point() : Math::point
- pointat() : Math::line
- PointLightEntity() : Graphics::PointLightEntity
- polar() : Math::polar
- Polygon() : Http::SvgPageWriter
- PolyLine() : Http::SvgPageWriter
- PoolArrayAllocator() : Memory::PoolArrayAllocator
- PoolLoadingResource() : Resources::PoolLoadingResource
- PoolResourceMapper() : Resources::PoolResourceMapper
- Pop() : Util::Stack<TYPE>
- Port() : IO::URI
- PrecomputeKeySliceValues() : CoreAnimation::AnimClip
- PreloadDialogTables() : Script::DialogManager
- PreloadTextures() : Graphics::MouseRenderer, Base::MouseRenderDeviceBase
- Prepare() : Base::ShaderVariableInstanceBase, Resources::ResourceManager
- PreSample() : Particles::EnvelopeCurve
- Present() : Direct3D9::D3D9RenderDevice
Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice

- PrimitiveGroup() : CoreGraphics::PrimitiveGroup
- print() : Debug::DebugFloat
- Print() : IO::Console, IO::ConsoleHandler
- print() : Debug::DebugFloat
- Print() : IO::HistoryConsoleHandler, OSX::OSXConsoleHandler
- print() : Debug::DebugFloat
- Print() : Win32::Win32ConsoleHandler, IO::LogFileConsoleHandler, IO::Console
- printHex() : Debug::DebugFloat
- ProcessWindowMessages() : Win32::Win32DisplayDevice, Base::DisplayDeviceBase
- project_screen_rh() : Math::sphere
- Property() : Game::Property
- PSSMUtil() : Lighting::PSSMUtil
- Ptr() : Ptr<TYPE>
- Push() : Util::Stack<TYPE>
- PushFlush() : Jobs::TPJobPort, Jobs::SerialJobPort, Jobs::TPJobPort, Base::JobPortBase
- PushJob() : Jobs::TPJobPort, Jobs::SerialJobPort, Jobs::TPJobPort, Base::JobPortBase
- PushJobChain() : Jobs::TPJobPort, Base::JobPortBase, Jobs::TPJobPort, Jobs::SerialJobPort
- PushJobCommand() : Jobs::TPWorkerThread
- PushJobSlices() : Jobs::TPJobThreadPool
- PushSync() : Base::JobPortBase, Jobs::SerialJobPort, Jobs::TPJobPort, Jobs::TPJobThreadPool
- PutBool() : Script::ActionReader
- PutClass() : Script::ActionReader
- PutEntity() : Script::ActionReader
- PutEvent() : Base::InputServerBase, CoreGraphics::RenderEventHandler, CoreGraphics::ThreadSafeDisplayEventHandler, Base::InputServerBase, CoreGraphics::ThreadSafeDisplayEventHandler, CoreGraphics::ThreadSafeRenderEventHandler, Base::InputServerBase, CoreGraphics::DisplayEventHandler
- PutFloat() : Script::ActionReader
- PutFloat4() : `Script::ActionReader`
- PutInt() : `Script::ActionReader`
- PutRequest() : `Http::HttpRequestHandler`
- PutString() : `Script::ActionReader`
- PutVector() : `Script::ActionReader`
- PutVersion() : `Script::ActionReader`
- q -

- quaternion() : Math::quaternion
- Query() : IO::URI
- Queue() : Util::Queue< TYPE >

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
- r -

- Rand() : Util::RandomNumberTable
- Read() : IO::MemoryStream, OSX::OSXFSWrapper, IO::Stream, System::Win32Environment, IO::Stream, Conditions::Condition, IO::Stream, IO::FileStream, Win360::Win360FSWrapper, IO::ZipFileStream, IO::ZipFileEntry, Actions::Action, Actions::ActionList, Actions::IfThenElseAction, Actions::SequenceAction, Conditions::And, Conditions::Condition, Conditions::Not, Conditions::Or, Actions::Action
- ReadAll() : IO::TextReader
- ReadAllLines() : IO::TextReader
- ReadBlob() : IO::BinaryReader
- ReadBool() : IO::BinaryReader
- ReadChar() : IO::TextReader, IO::BinaryReader
- ReadDInputMouse() : Win32::Win32InputServer
- ReadDouble() : IO::BinaryReader
- ReadFloat() : IO::BinaryReader
- ReadFloat2() : IO::BinaryReader
- ReadFloat4() : IO::BinaryReader
- ReadFloatFromNormalizedUByte2() : IO::BinaryReader
- ReadFloatFromUnsignedNormalizedUByte2() : IO::BinaryReader
- ReadGuid() : IO::BinaryReader
- ReadInt() : System::Win32Registry, IO::BinaryReader
- `ReadLine()`: `IO::TextReader`
- `ReadMatrix44()`: `IO::BinaryReader`
- `ReadMessage()`: `Messaging::MessageReader`
- `ReadPoint()`: `IO::BinaryReader`
- `ReadRawData()`: `IO::BinaryReader`
- `ReadRequest()`: `Http::HttpRequestReader`
- `ReadResponse()`: `Http::HttpResponseReader`
- `ReadShort()`: `IO::BinaryReader`
- `ReadString()`: `System::Win32Registry`, `IO::BinaryReader`
- `ReadTexturePoolFromXML()`: `Resources::PoolResourceMapper`
- `ReadUChar()`: `IO::BinaryReader`
- `ReadUInt()`: `IO::BinaryReader`
- `ReadUShort()`: `IO::BinaryReader`
- `ReadValue()`: `Math::Extrapolator< TYPE >`
- `ReadVector()`: `IO::BinaryReader`
- `Realloc()`: `OSX::OSXHeap`, `Win360::Win360Heap`, `Util::Array< TYPE >`
- `reciprocal()`: `Math::float4`
- `Rect()`: `Http::SvgPageWriter`
- `rectangle()`: `Math::rectangle< TYPE >`
- `RecurseCreateNodeInstanceHierarchy()`: `Models::ModelNode`
- `Recv()`: `Net::MessageClient`, `Net::MessageClientConnection`, `Net::StdTcpClient`, `Net::StdTcpClientConnection`, `Net::StdTcpServer`, `Win360::Win360Socket`, `Net::StdTcpClient`, `Net::StdTcpClientConnection`, `Win360::Win360Socket`, `Net::StdTcpServer`
- `RecvFrom()`: `Win360::Win360Socket`
- `Ref()`: `Threading::ObjectRef`
- `RefCounted()`: `Core::RefCounted`
- `reflect()`: `Math::float4`, `Math::matrix44`
- `Register()`: `Core::Factory`
- `RegisterAnimEventHandler()`: `Animation::AnimEventServer`
- `RegisterDebugCounter()`: `Debug::DebugServer`
- `RegisterDebugTimer()`: `Debug::DebugServer`
- `RegisterEntity()`: `Visibility::VisibilityChecker`
- `RegisterMessage()`: `Messaging::Port`
- `RegisterPropertyCallback()`: `Game::Entity`
- `RegisterRenderModule()`: `Graphics::GraphicsServer`
- `RegisterRTPlugin()`: `RenderModules::RTPluginRegistry`
- `RegisterSoftwareSkinnedMesh()`: `Base::SkinnedMeshRendererBase`
- `RegisterUnmanagedResource()`: `Resources::ResourceManager`
- `RegisterUriScheme()`: `IO::SchemeRegistry`
- `Release()`: `Core::RefCounted`, `Util::SimpleTree< VALUETYPE >::Node`, `Core::RefCounted`
- `ReleaseKeyboardCapture()`: `Base::InputServerBase`
- `ReleaseMouseCapture()`: `Base::InputServerBase`
- `ReleaseResources()`: `Resources::ResourceManager`
- `ReloadScripts()`: `Script::ScriptManager`
- `Remove()`: `Util::List< TYPE >`
- `RemoveAllEntities()`: `InternalGraphics::InternalStage`
- `RemoveAllHandlers()`: `Messaging::Port`
- `RemoveAllLoaders()`: `BaseGameFeature::LoaderServer`
- `RemoveAllMappers()`: `Resources::ResourceManager`
- `RemoveBack()`: `Util::List< TYPE >`
- `RemoveDisplayEventHandler()`: `Graphics::Display`
- `RemoveEntity()`: `InternalGraphics::InternalStage`, `BaseGameFeature::EntityManager`
- `RemoveEntityFromTriggered()`: `BaseGameFeature::EntityManager`
- `RemoveEntityImmediate()`: `BaseGameFeature::EntityManager`
- `RemoveEntityLoader()`: `BaseGameFeature::LoaderServer`
- `RemoveEventHandler()`: `Base::DisplayDeviceBase`, `Base::RenderDeviceBase`
- `RemoveFront()`: `Util::List< TYPE >`
- `RemoveGameFeature()`: `Game::GameServer`
- `RemoveHandler()`: `Messaging::HandlerThreadBase`, `Messaging::Port`, `Messaging::HandlerThreadBase`, `Messaging::Port`, `Messaging::HandlerThreadBase`, `Messaging::Port`, `Messaging::AsyncPort`, `Messaging::Port`, `IO::Console`, `Messaging::Port`, `Messaging::AsyncPort`, `Messaging::Port`, `Messaging::HandlerThreadBase`, `Messaging::Port`, `Messaging::AsyncPort`, `Messaging::Port`, `Messaging::AsyncPort`, `Messaging::Port`, `Messaging::AsyncPort`, `Messaging::Port`
- `RemoveInputHandler()`: `Base::InputServerBase`
- RemoveInvalidAttachments:
  `InternalGraphics::AttachmentServer`
- RemoveManager:
  `Game::FeatureUnit`
- RemoveMapper:
  `Resources::ResourceManager`
- RemoveNode:
  `Models::Model`
- RemoveNodeInstance:
  `Models::ModelInstance`
- RemoveNullEntriesFromArrays:
  `BaseGameFeature::EntityManager`
- RemovePort:
  `Messaging::Dispatcher`
- RemoveRenderEventHandler:
  `Graphics::Display`
- RemoveRequestHandler:
  `Http::HttpServerProxy, Http::HttpServer`
- RemoveSkin:
  `Characters::CharacterSkinSet`
- RemoveTimeSource:
  `BaseGameFeature::TimeManager`
- RemoveVisibilityContext:
  `Visibility::VisibilityClusterSystem, Visibility::VisibilityQuadtree, Visibility::VisibilitySystemBase, Visibility::VisibilityBoxSystem`
- RemoveVisibilitySystem:
  `Visibility::VisibilityChecker`
- RenameLevel:
  `BaseGameFeature::CategoryManager`
- Render:
  `Characters::CharacterSkinNodeInstance, Models::ModelNodeInstance, Models::ShapeNodeInstance, InternalGraphics::InternalView, Models::ModelNodeInstance, Frame::FrameShader, Frame::FramePostEffect, Frame::FramePassBase, Models::ModelNodeInstance, Particles::ParticleSystemNodeInstance, Models::ModelNodeInstance, Frame::FrameBatch, Frame::FramePass, Models::ModelNodeInstance, Direct3D9::D3D9ParticleSystemInstance`
- RenderApplication:
  `App::RenderApplication`
- RenderDebug:
  `Models::ModelNodeInstance, Models::TransformNodeInstance, Particles::ParticleSystemNodeInstance, InternalGraphics::InternalView, Models::ModelNodeInstance, Models::ModelNodeInstance, Models::TransformNodeInstance, Characters::CharacterInstance, Characters::CharacterSkeletonInstance, Characters::CharacterNodeInstance, Models::TransformNodeInstance`
- RenderDebugVisualization() : PhysicsFeature::TriggerProperty
- RenderDevice() : CoreGraphics::RenderDevice
- RenderDeviceBase() : Base::RenderDeviceBase
- RenderEvent() : CoreGraphics::RenderEvent
- RenderEventHandler() : CoreGraphics::RenderEventHandler
- RenderGlobalLight() : Lighting::LightPrePassServer
- RenderLights() : Lighting::LightServerBase, Lighting::LightPrePassServer, Lighting::LightServerBase
- RenderModule() : RenderModules::RenderModule
- RenderParticleSystem() : Base::ParticleRendererBase
- RenderPointers() : Base::MouseRenderDeviceBase
- RenderPointLights() : Lighting::LightPrePassServer
- RenderShape() : CoreGraphics::RenderShape
- RenderSpotLights() : Lighting::LightPrePassServer
- RenderTargetBase() : Base::RenderTargetBase
- ReplaceChars() : Util::String
- ReplaceIllegalFilenameChars() : Util::String
- RequestResourceForLoading() : Resources::ResourceManager
- RequestState() : App::GameApplication
- Reserve() : Util::Array<TYPE>, Util::Queue<TYPE>, Util::Array<TYPE>, Util::Dictionary<KEYTYPE, VALUETYPE>, Util::Blob, Util::Queue<TYPE>, Util::String
- ReserveFragments() : Characters::CharacterSkinNode
- ReserveSkinLists() : Characters::CharacterSkinLibrary
- ReserveSkins() : Characters::CharacterSkinLibrary
- Reset() : Resources::ResourceLoader, Resources::D3D9TextureStreamer, BaseGameFeature::TimeSource, Resources::ResourceLoader, Resources::D3D9TextureStreamer, Resources::ResourceLoader, Util::Array<TYPE>, Win360::Win360Event, RenderUtil::MayaCameraUtil, Resources::ResourceLoader, Base::InputServerBase, Resources::ResourceLoader, BaseGameFeature::TimeSource, Base::InputServerBase, BaseGameFeature::TimeSource, Math::Extrapolator<TYPE>, Resources::ResourceLoader, Win360::Win360Event, Debug::DebugCounter, Resources::ResourceLoader, BaseGameFeature::TimeSource, Win360::Win360Timer,
Base::InputServerBase, Resources::ResourceLoader, Util::RingBuffer< TYPE >, Util::Array< TYPE >, Models::VisResolveContainer< TYPE >, Resources::PoolResourceMapper, Math::Extrapolator< TYPE >, Resources::ResourceLoader
- ResetAccum() : Debug::DebugTimer
- ResetAll() : BaseGameFeature::TimeManager
- ResetFeatureBits() : Base::ShaderServerBase
- ResetModelNodeInstanceIndex() : Models::ModelServer
- ResetScreenSpaceStats() : Models::ModelNode
- ResetTime() : FrameSync::FrameSyncTimer
- Resize() : Util::FixedArray< TYPE >
- ResolveAssigns() : IO::AssignRegistry
- ResolveAssignsInString() : IO::AssignRegistry
- ResolveDepthBuffer() : Base::RenderTargetBase
- ResolveVisibleLights() : InternalGraphics::InternalView
- ResolveVisibleModelNodeInstances() : InternalGraphics::InternalView
- Resource() : Resources::Resource
- ResourceBase() : Base::ResourceBase
- ResourceDictionary() : Resources::ResourceDictionary
- ResourceLoader() : Resources::ResourceLoader
- ResourceManager() : Resources::ResourceManager
- ResourceMapper() : Resources::ResourceMapper
- ResourceSaver() : Resources::ResourceSaver
- ReuseMips() : Resources::D3D9TextureStreamer
- RingBuffer() : Util::RingBuffer< TYPE >
- Root() : Util::SimpleTree< VALUETYPE >
- rotationaxis() : Math::matrix44, Math::quaternion
- rotationmatrix() : Math::quaternion, Math::matrix44
- rotationquaternion() : Math::matrix44
- rotationx() : Math::matrix44
- rotationy() : Math::matrix44
- rotationyawpitchroll() : Math::matrix44, Math::quaternion
- rotationz() : Math::matrix44
- Row() : Util::QuadTree< TYPE >::Node
-.RowIndex() : BaseGameFeature::CategoryManager::Entry
- RTPlugin() : RenderModules::RTPlugin
- RTPluginRegistry() : RenderModules::RTPluginRegistry
- Rtti() : Core::Rtti
- Run() : Visibility::VisibilityQuery, App::Application, App::GameApplication, App::RenderApplication, App::Application
- Running() : Win360::Win360Timer
- RunThroughHandlerThread() : Messaging::RunThroughHandlerThread
- s -

- SafeFlag() : Threading::SafeFlag
- SafePriorityQueue() : Threading::SafePriorityQueue<PRITYPE, TYPE>
- SafeQueue() : Threading::SafeQueue< TYPE >
- Sample() : CoreAnimation::AnimUtil, Particles::EnvelopeCurve
- Save() : Script::Task, Resources::Resource, BaseGameFeature::UserProfile
- SaveAttributes() : BaseGameFeature::GlobalAttrsManager
- SaveDialogs() : Script::DialogManager
- SaveDialogTakeStates() : Script::DialogManager
- SaveScreenshot() : Direct3D9::D3D9RenderDevice, Base::RenderDeviceBase
- scale() : Math::matrix44
- scaling() : Math::matrix44
- Scheme() : IO::URI
- SchemeRegistry() : IO::SchemeRegistry
- ScriptFeatureUnit() : ScriptFeature::ScriptFeatureUnit
- ScriptManager() : Script::ScriptManager
- Seek() : IO::Stream, OSX::OSXFSWrapper, IO::FileStream, IO::MemoryStream, IO::Stream, Win360::Win360FSWrapper, IO::ZipFileStream, IO::Stream
- SegmentedGfxUtil() : GraphicsFeature::SegmentedGfxUtil
- SelectActiveVariation() : Base::ShaderInstanceBase,
Direct3D9::D3D9ShaderInstance
- Send() : Messaging::AsyncPort, Win360::Win360Socket, Net::StdTcpClient, Net::StdTcpClientConnection, Win360::Win360Socket, Messaging::Port, Messaging::AsyncPort, Messaging::Port, Messaging::AsyncPort, Net::MessageClient, Net::MessageClientConnection, Net::StdTcpClient, Messaging::Port, Net::StdTcpClientConnection, Messaging::Port, Messaging::AsyncPort, Net::StdTcpClientConnection
- SendBatched() : Graphics::GraphicsInterface
- SendCreateMsg() : Graphics::GraphicsEntity
- SendMsg() : Graphics::GraphicsEntity
- SendStop() : PhysicsFeature::ActorPhysicsProperty
- SendSync() : Game::Entity
- SendTo() : Win360::Win360Socket
- SendWait() : Messaging::AsyncPort
- SequenceAction() : Actions::SequenceAction
- SerialJob() : Jobs::SerialJob
- SerialJobFuncDesc() : Jobs::SerialJobFuncDesc
- SerialJobPort() : Jobs::SerialJobPort
- SerialJobSystem() : Jobs::SerialJobSystem
- set() : Math::float4
- Set() : Threading::SafeFlag
- set() : Math::float4, OSX::OSXThreadLocalPtr< TYPE >, Math::bbox, Math::float2
- Set() : Debug::DebugCounter
- set() : Math::frustum, Math::line, Math::polar, Math::rectangle< TYPE >
- Set() : IO::MediaType, Util::Blob
- set() : Math::sphere
- Set() : Util::FixedTable< TYPE >
- set() : Math::float4
- Set() : Util::String
- set() : Math::matrix44
- Set() : IO::URI
- set() : Math::plane, Math::point, Math::quaternion, Math::vector
- Set() : IO::XmlWriter
- `set_a()` : `Math::plane`
- `set_b()` : `Math::plane`
- `set_c()` : `Math::plane`
- `set_d()` : `Math::plane`
- `set_position()` : `Math::matrix44`
- `set_w()` : `Math::float4`, `Math::quaternion`
- `set_x()` : `Math::float4`, `Math::quaternion`
- `set_xaxis()` : `Math::matrix44`
- `set_y()` : `Math::float4`, `Math::quaternion`
- `set_yaxis()` : `Math::matrix44`
- `set_z()` : `Math::float4`, `Math::quaternion`
- `set_zaxis()` : `Math::matrix44`
- `SetAbsMousePos()` : `Input::InputEvent`
- `SetAccess()` : `Base::ResourceBase`, `CoreGraphics::MemoryMeshLoader`, `Win360::D3D9StreamMeshLoader`, `Base::ResourceBase`, `Win360::D3D9StreamMeshLoader`, `Base::ResourceBase`
- `SetAccessMode()` : `IO::Stream`
- `SetAccessPattern()` : `IO::Stream`
- `SetActionBlock()` : `Script::DialogTake`
- `SetActionList()` : `Actions::ActionList`, `Script::DialogTake`
- `SetActionRef()` : `Script::DialogTake`
- `SetActive()` : `CoreAnimation::AnimCurve`
- `SetActiveShaderInstance()` : `Base::ShaderServerBase`
- `SetActiveStatus()` : `PhysicsFeature::TriggerProperty`
- `SetActiveVariation()` : `Characters::CharacterVariationSet`
- `SetAdapter()` : `Graphics::DisplaySettings`, `Base::DisplayDeviceBase`
- `SetAddress()` : `Win360::Win360Socket`, `Net::StdTcpServer`, `Win360::Win360Socket`
- `SetAllParams()` : `Graphics::GlobalLightEntity`
- `SetAlpha()` : `CoreGraphics::MousePointer`
- `SetAlwaysOnTop()` : `Graphics::DisplaySettings`, `Base::DisplayDeviceBase`
- `SetAmbientLightColor()` : `Graphics::GlobalLightEntity`, `Lighting::InternalGlobalLightEntity`
- `SetAnimationResourceId()` : `Characters::CharacterNode`
- `SetAnimEvent()` : `Animation::AnimEventInfo`
- `SetAnimJobName()` : `Animation::AnimEventInfo`
- `SetAntiAliasQuality()`: `Graphics::DisplaySettings`, `Base::DisplayDeviceBase`, `Base::RenderTargetBase`, `Base::DisplayDeviceBase`, `Base::RenderTargetBase`
- `SetAppID()`: `App::Application`
- `SetAppName()`: `Interface::InterfaceHandlerBase`, `Core::CoreServer`, `Interface::InterfaceHandlerBase`
- `SetAppName()`: `App::Application`
- `SetAppVersion()`: `App::Application`
- `SetArchiveFileSystemEnabled()`: `IO::IoServer`
- `SetAspectRatio()`: `CoreGraphics::DisplayMode`
- `SetAssign()`: `IO::AssignRegistry`
- `SetAsyncEnabled()`: `Resources::Resource`, `Resources::ResourceMapper`, `Resources::Resource`, `Resources::ResourceMapper`
- `SetAttr()`: `Game::Entity`
- `SetAttrValue()`: `Game::Entity`
- `SetAutoManaged()`: `Resources::ManagedResource`
- `SetBackLightColor()`: `Graphics::GlobalLightEntity`, `Lighting::InternalGlobalLightEntity`
- `SetBackLightOffset()`: `Graphics::GlobalLightEntity`, `Lighting::InternalGlobalLightEntity`
- `SetBaseIndex()`: `CoreGraphics::PrimitiveGroup`
- `SetBaseTime()`: `Animation::AnimJob`
- `SetBaseVertex()`: `CoreGraphics::PrimitiveGroup`
- `SetBit()`: `Util::BitField< NUMBITS >`
- `SetBlendWeight()`: `Animation::AnimJob`
- `SetBlob()`: `Util::Variant`, `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`
- `SetBlobArray()`: `Util::Variant`
- `SetBlocking()`: `Net::StdTcpClient`, `Win360::Win360Socket`, `Net::StdTcpClient`, `Win360::Win360Socket`
- `SetBlueprintsFilename()`: `BaseGameFeature::FactoryManager`
- `SetBool()`: `Base::ShaderVariableBase`, `Base::ShaderVariableInstanceBase`, `Direct3D9::D3D9ShaderVariable`, `Util::String`, `Util::Variant`, `Direct3D9::D3D9ShaderVariable`, `Base::ShaderVariableInstanceBase`, `Particles::EmitterAttrs`, `BaseGameFeature::UserProfile`, `BaseGameFeature::GlobalAttrsManager`, `IO::XmlWriter`, ..
Game::Entity
- SetBoolArray() : Base::ShaderVariableBase, Direct3D9::D3D9ShaderVariable, Util::Variant, Direct3D9::D3D9ShaderVariable
- SetBoundingBox() : Models::ModelNode, CoreGraphics::PrimitiveGroup, Models::ModelNode, Models::Model, Models::ModelNode
- SetBroadcast() : Win360::Win360Socket
- SetCameraEntity() : Graphics::View, InternalGraphics::InternalView, Lighting::PSSMUtil
- SetCameraFocusEntity() : BaseGameFeature::FocusManager
- SetCameraFocusToNextEntity() : BaseGameFeature::FocusManager
- SetCameraSettings() : Graphics::CameraEntity, InternalGraphics::InternalCameraEntity
- SetCanvasDimensions() : Http::SvgPageWriter
- SetCapacity() : Util::RingBuffer< TYPE >
- SetCastShadowsThisFrame() : Lighting::InternalAbstractLightEntity
- SetCategory() : CoreAnimation::AnimEvent
- SetCategoryName() : Animation::AnimEventHandlerBase
- SetChar() : Input::InputEvent
- SetCharPtr() : Util::String
- SetClearColor() : Base::RenderTargetBase, Frame::FramePassBase, Base::MultipleRenderTargetBase, Base::RenderTargetBase, Frame::FramePassBase, Base::MultipleRenderTargetBase
- SetClearDepth() : Base::RenderTargetBase, Frame::FramePassBase, Base::MultipleRenderTargetBase, Base::RenderTargetBase, Frame::FramePassBase, Base::MultipleRenderTargetBase
- SetClearFlags() : Frame::FramePassBase, Base::RenderTargetBase, Frame::FramePassBase, Base::RenderTargetBase
- SetClearStencil() : Base::RenderTargetBase,
Frame::FramePassBase, Base::MultipleRenderTargetBase,
Base::RenderTargetBase, Frame::FramePassBase,
Base::MultipleRenderTargetBase
- SetClientConnectionClass() : Net::StdTcpServer
- SetClipName() : Animation::PlayClipJob
- SetCloseText() : Script::Task
- SetCmdLineArgs() : App::Application, Game::FeatureUnit,
  App::Application, Game::FeatureUnit
- SetColor() : Lighting::InternalAbstractLightEntity,
  Graphics::AbstractLightEntity,
  Lighting::InternalAbstractLightEntity,
  Graphics::AbstractLightEntity
- SetColorBufferFormat() : Base::RenderTargetBase
- SetCompanyName() : Interface::InterfaceHandlerBase,
  App::Application, Interface::InterfaceHandlerBase,
  Core::CoreServer, Interface::InterfaceHandlerBase
- SetCondition() : Actions::IfThenElseAction, Conditions::Not,
  Script::DialogTake
- SetContent() : Http::HttpResponseWriter
- SetConversation() : Script::DialogDesc
- SetCoreId() : Win360::Win360Thread, OSX::OSXThread,
  Win360::Win360Thread
- SetCurrentEntitiesInTrigger() : PhysicsFeature::TriggerProperty
- SetCurrentNodeIndex() : IO::BXmlLoaderUtil
- SetCurrentTake() : Script::Dialog
- SetCurVertexPtr() : Direct3D9::D3D9ParticleRenderer
- SetCurveType() : CoreAnimation::AnimCurve
- SetD3D9IndexBuffer() : Win360::D3D9IndexBuffer
- SetD3D9VertexBuffer() : Win360::D3D9VertexBuffer
- SetD3D9VertexDeclaration() : Win360::D3D9VertexLayout
- SetDataValid() : Net::DebugPacket
- SetDay() : Base::CalendarTimeBase
- SetDebugHudEnabled() : Animation::AnimSequencer
- SetDebugTextEnabled() : BaseGameFeature::LoaderServer
- SetDefaultPoolScheduler() : Resources::PoolResourceMapper
- SetDefaultRenderTarget() : Base::RenderTargetBase
- SetDefaultView() : Graphics::GraphicsServer,
  InternalGraphics::InternalGraphicsServer
- SetDeferred() : Messaging::Message
- SetDeferredHandled() : Messaging::Message
- SetDepth() : Base::TextureBase
- SetDesc() : Http::HttpRequestHandler
- SetDescription() : Script::InfoLog, CoreGraphics::AdapterInfo
- SetDeviceId() : CoreGraphics::AdapterInfo
- SetDeviceIndex() : Input::InputEvent
- SetDeviceName() : CoreGraphics::AdapterInfo
- SetDialogLockState() : Script::DialogManager
- SetDialogTakeState() : Script::DialogManager
- SetDisplayMode() : Base::DisplayDeviceBase
- SetDisplayModeSwitchEnabled() : Graphics::DisplaySettings, Base::DisplayDeviceBase
- SetDriverName() : CoreGraphics::AdapterInfo
- SetDriverVersionHighPart() : CoreGraphics::AdapterInfo
- SetDriverVersionLowPart() : CoreGraphics::AdapterInfo
- SetDuration() : Animation::AnimJob
- SetElement() : Util::QuadTree<TYPE>::Node
- SetElseBlock() : Actions::IfThenElseAction
- SetEmitterAttrs() : Particles::ParticleSystemNode
- SetEmitterMeshResourceId() : Particles::ParticleSystemNode
- SetEmote() : Script::DialogTake
- SetEnabled() : PhysicsFeature::PhysicsProperty, GraphicsFeature::LightFlickerUtil, PhysicsFeature::PhysicsProperty
- SetEnqueueMode() : Animation::AnimJob
- SetEntitiesLastFrameInTrigger() : PhysicsFeature::TriggerProperty
- SetEntity() : FSM::StateMachine, Actions::Action, Actions::ActionList, Conditions::Or, Game::Property, Actions::Action, Conditions::Condition, Actions::SequenceAction, Game::Property, Actions::IfThenElseAction, Conditions::And, Conditions::Condition, Conditions::Not
- SetEntityId() : Animation::AnimEventInfo
- SetEntityMask() : Visibility::VisibilityQuery
- SetEntryDirect() : Util::SparseTable<TYPE>
- SetEnvelope() : Particles::EmitterAttrs
- SetEnvEntityTransform() : BaseGameFeature::EnvEntityManager
- SetExclusiveTag() : `Animation::AnimJob`
- SetFactor() : `BaseGameFeature::TimeSource`
- SetFadeInTime() : `Animation::AnimJob`
- SetFadeOutTime() : `Animation::AnimJob`
- SetFailedText() : `Script::Task`
- SetFeatureBits() : `Base::ShaderServerBase`
- SetFeatureMask() : `Base::ShaderVariationBase`
- SetFileName() : `StateObjectFeature::StateProperty`
- SetFileWriteTime() : `IO::IoServer`, `OSX::OSXFSWrapper`, `Win360::Win360FSWrapper`
- SetFirstKeyIndex() : `CoreAnimation::AnimCurve`
- SetFloat() : `Base::ShaderVariableBase`, `Base::ShaderVariableInstanceBase`, `Util::String`, `Direct3D9::D3D9ShaderVariable`, `Util::Variant`, `Direct3D9::D3D9ShaderVariable`, `Base::ShaderVariableInstanceBase`, `Particles::EmitterAttrs`, `BaseGameFeature::UserProfile`, `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`, `IO::XmlWriter`
- SetFloat2() : `Util::String`, `IO::XmlWriter`
- SetFloat4() : `Base::ShaderVariableBase`, `Base::ShaderVariableInstanceBase`, `Direct3D9::D3D9ShaderVariable`, `Util::String`, `Util::Variant`, `Direct3D9::D3D9ShaderVariable`, `Base::ShaderVariableInstanceBase`, `BaseGameFeature::UserProfile`, `BaseGameFeature::GlobalattrsManager`, `Game::Entity`, `IO::XmlWriter`
- SetFloat4Array() : `Base::ShaderVariableBase`, `Util::Variant`, `Direct3D9::D3D9ShaderVariable`
- SetFloatArray() : `Direct3D9::D3D9ShaderVariable`, `Base::ShaderVariableBase`, `Util::Variant`, `Direct3D9::D3D9ShaderVariable`
- SetFocalLength() : `Base::TransformDeviceBase`
- SetFocusEntity() : `BaseGameFeature::FocusManager`
- SetFocusToNextEntity() : `BaseGameFeature::FocusManager`
- SetFormat() : `Base::StreamTextureSaverBase`
- SetFragment() : `IO::URI`
- SetFrameId() : `Resources::ManagedResource`
- SetFrameShader() : InternalGraphics::InternalView
- SetFrequency() : GraphicsFeature::LightFlickerUtil
- SetFromByteOrder() : System::ByteOrder
- SetFromHexString() : Util::FourCC
- SetFromUInt() : Util::FourCC
- SetFullscreen() : Base::DisplayDeviceBase, Graphics::DisplaySettings, Base::DisplayDeviceBase
- SetGlobalMatrix() : Shared::CharJointInfo
- SetGraphicsEntities() : Visibility::VisibilityContainer
- SetGroup() : Script::DialogDesc
- SetGuid() : Util::Variant, BaseGameFeature::GlobalAttrsManager, Game::Entity, CoreGraphics::AdapterInfo, Script::DialogDesc, Script::DialogTake
- SetGuidArray() : Util::Variant
- SetHandled() : Messaging::Message
- SetHandlerThread() : Messaging::AsyncPort
- SetHeight() : Base::TextureBase, CoreGraphics::DisplayMode, Base::RenderTargetBase, Base::TextureBase, Base::RenderTargetBase
- SetHighFrequencyVibrator() : Base::GamePadBase
- SetHistorySize() : IO::HistoryConsoleHandler
- SetHost() : IO::URI
- SetHostName() : Win360::Win360IpAddress
- SetHotspot() : CoreGraphics::MousePointer
- SetHour() : Base::CalendarTimeBase
- SetIconName() : Graphics::DisplaySettings, Base::DisplayDeviceBase
- SetId() : Script::DialogDesc, Script::DialogTake
- SetIndexBuffer() : Base::MeshBase, Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice, CoreGraphics::MemoryMeshLoader, Base::MeshBase, Direct3D9::D3D9RenderDevice
- SetIndexType() : Base::IndexBufferBase
- SetInputFocusEntity() : BaseGameFeature::FocusManager
- SetInputFocusToNextEntity() : BaseGameFeature::FocusManager
- SetInstanceEntity() : BaseGameFeature::CategoryManager
- SetInt() : Base::ShaderVariableBase, IO::XmlWriter,
Base::ShaderVariableInstanceBase, Util::String,
Direct3D9::D3D9ShaderVariable, Util::Variant,
Direct3D9::D3D9ShaderVariable,
Base::ShaderVariableInstanceBase, Particles::EmitterAttrs,
BaseGameFeature::UserProfile,
BaseGameFeature::GlobalAttrsManager, Game::Entity

- SetIntArray() : Base::ShaderVariableBase, Util::Variant,
  Direct3D9::D3D9ShaderVariable
- SetIntensityAmplitude() : GraphicsFeature::LightFlickerUtil
- SetInViewSpace() : Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode,
  Models::TransformNodeInstance
- SetJointName() : Shared::CharJointInfo
- SetKeepAlive() : Win360::Win360Socket
- SetKey() : Input::InputEvent
- SetKeyDuration() : CoreAnimation::AnimClip
- SetKeyStride() : CoreAnimation::AnimClip
- SetLevelName() : BaseGameFeature::GameStateHandler
- SetLightDir() : Lighting::PSSMUtil
- SetLightEntity() : GraphicsFeature::LightFlickerUtil
- SetLightingMode() : Frame::FrameBatch
- SetLightType() : Lighting::InternalAbstractLightEntity,
  Graphics::AbstractLightEntity,
  Lighting::InternalAbstractLightEntity,
  Graphics::AbstractLightEntity
- SetLoader() : Resources::Resource
- SetLocalBoundingBox() :
  InternalGraphics::InternalGraphicsEntity
- SetLocalizeDialog() : Script::DialogManager
- SetLocalMatrix() : Shared::CharJointInfo
- SetLocalPath() : IO::URI
- SetLocked() : Script::DialogDesc
- SetLockedToViewer() : Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode,
- SetLogFileEnabled() : **App::RenderApplication**
- SetLogLevel() : **Script::InfoLog**
- SetLoopCount() : **Animation::PlayClipJob**
- SetLowFrequencyVibrator() : **Base::GamePadBase**
- SetMainRenderTarget() : **Frame::FrameShader**
- SetManagedResource() : **Resources::LoadingResource**
- SetManagedResourceClass() : **Resources::SimpleResourceMapper**
- SetMapper() : **Resources::ResourceScheduler**, **Resources::TexturePoolMapperScheduler**
- SetMatrix() : **Base::ShaderVariableBase**, **Base::ShaderVariableInstanceBase**, **Direct3D9::D3D9ShaderVariable**, **Base::ShaderVariableInstanceBase**, **Direct3D9::D3D9ShaderVariable**
- SetMatrix44() : **Util::String**, **Util::Variant**, **Game::Entity**, **IO::XmlWriter**, **BaseGameFeature::GlobalAttrsManager**
- SetMatrix44Array() : **Util::Variant**
- SetMatrixArray() : **Base::ShaderVariableBase**, **Direct3D9::D3D9ShaderVariable**
- SetMaxDistance() : **Models::TransformNode**
- SetMaxNumLocalPlayers() : **Base::InputServerBase**
- SetMaxProgressValue() : **BaseGameFeature::LoaderServer**
- SetMaxTriggerDistance() : **BaseGameFeature::EntityManager**
- SetMediaType() : **IO::Stream**
- SetMemoryMappingEnabled() : **IO::BinaryReader**, **IO::BinaryWriter**
- SetMemoryOffset() : **Base::RenderTargetBase**
- SetMeshResourceId() : **Models::ShapeNode**
- SetMeshResourceLoader() : **Models::ShapeNode**
- SetMethod() : **Http::HttpRequestWriter**, **Http::HttpRequest**
- SetMilliSecond() : **Base::CalendarTimeBase**
- SetMinDistance() : **Models::TransformNode**
- SetMinute() : **Base::CalendarTimeBase**
- SetMipLevel() : **Base::StreamTextureSaverBase**
- SetMipMapsEnabled() : **Base::RenderTargetBase**
- SetModelEntity() : **Models::ModelInstance**
- SetModelResourceMapper() : **Models::ModelServer**
- SetModelTransform() : **Base::TransformDeviceBase**
- SetMonth() : `Base::CalendarTimeBase`
- SetMountStandardArchivesEnabled() : `App::RenderApplication`
- SetMouseButton() : `Input::InputEvent`
- SetMouseMovement() : `RenderUtil::MayaCameraUtil`
- SetMRTIndex() : `Base::RenderTargetBase`
- SetMultipleRenderTarget() : `Frame::FramePassBase, InternalGraphics::InternalView, Frame::FramePassBase`
- SetMyThreadName() : `Win360::Win360Thread`
- SetName() : `Http::HttpRequestHandler, Win360::Win360Thread, OSX::OSXThread, Base::ShaderVariableBase, Win360::Win360Thread, Animation::AnimJob, Win360::Win360Thread, Models::ModelNode, Win360::Win360Thread, Models::ModelNode, FSM::State, Models::ModelNode, Base::ShaderVariationBase, Frame::FramePassBase, Frame::FrameShader, InternalGraphics::InternalStage, InternalGraphics::InternalView, Models::ModelNode, Base::ShaderVariableBase, Base::ShaderVariationBase, Animation::AnimJob, Base::ShaderVariationBase, Http::HttpRequestHandler, Characters::CharacterSkinList, Http::HttpRequestHandler, Frame::FramePassBase, Win360::Win360Thread, CoreAnimation::AnimClip, BaseGameFeature::UserProfile, CoreAnimation::AnimEvent, Win360::Win360Thread, Http::HttpRequestHandler, FSM::StateMachine`
- SetNodeFilter() : `Frame::FrameBatch`
- SetNoDelay() : `Win360::Win360Socket`
- SetNormMousePos() : `Input::InputEvent`
- SetNumCurves() : `CoreAnimation::AnimClip`
- SetNumIndices() : `Base::IndexBufferBase, CoreGraphics::PrimitiveGroup, Base::IndexBufferBase`
- SetNumKeys() : `CoreAnimation::AnimClip`
- SetNumMipLevels() : `Base::TextureBase`
- SetNumPasses() : `Base::ShaderVariationBase`
- SetNumVertices() : `Base::VertexBufferBase, CoreGraphics::PrimitiveGroup, Base::VertexBufferBase`
- SetObject() : `Util::Variant`
- SetObserver() : `Visibility::VisibilityQuery`
- setoffset() : `Math::transform44`
- `SetOffsetMatrix()` : `Models::TransformNodeInstance`
- `SetOpenText()` : `Script::Task`
- `SetOrbitButton()` : `RenderUtil::MayaCameraUtil`
- `SetOrbiting()` : `RenderUtil::MayaCameraUtil`
- `SetOrientation()` : `CoreGraphics::MousePointer`
- `SetOverrideDefaultRenderTarget()` : `Base::RenderDeviceBase`
- `SetOverrideRootDirectory()` : `App::RenderApplication`
- `SetPanButton()` : `RenderUtil::MayaCameraUtil`
- `SetPanning()` : `RenderUtil::MayaCameraUtil`
- `SetParent()` : `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`, `Models::ModelNode`, `Models::ModelNodeInstance`
- `SetParentKey()` : `Script::Task`
- `SetParentWindow()` : `Graphics::Display`, `Base::DeviceBase`
- `SetPerfHUDEnabled()` : `Debug::DebugGraphicsHandler`
- `SetPixelFormat()` : `Base::TextureBase`, `CoreGraphics::DisplayMode`, `Base::TextureBase`
- `SetPlaceholder()` : `Resources::ManagedResource`
- `SetPlaceholderResourceId()` : `Resources::ResourceMapper`
- `SetPlayerIndex()` : `Base::GamePadBase`
- `SetPort()` : `Http::HttpServer`, `IO::URI`, `Win360::Win360IpAddress`
- `SetPosition()` : `Models::TransformNodeInstance`, `Models::TransformNode`, `Models::TransformNodeInstance`, `Models::TransformNode`, `Models::TransformNode`, `CoreGraphics::MousePointer`, `Models::TransformNode`
- `setPosition()` : `Math::transform44`
- `SetPosition()` : `Models::TransformNodeInstance`, `Models::TransformNode`
- `SetPositionAmplitude()` : `GraphicsFeature::LightFlickerUtil`
- `SetPostInfinityType()` : `CoreAnimation::AnimClip`
- `SetPreInfinityType()` : `CoreAnimation::AnimClip`
- `SetPrimitiveGroup()` : `Base::RenderDeviceBase`
- `SetPrimitiveGroupIndex()` : `Models::ShapeNode`, `Particles::ParticleSystemNode`
- SetPrimitiveGroups() : Base::MeshBase, CoreGraphics::MemoryMeshLoader, Base::MeshBase
- SetPrimitiveTopology() : CoreGraphics::PrimitiveGroup
- SetPriority() : Win360::Win360Thread, OSX::OSXThread, Resources::LoadingResource, Win360::Win360Thread, Resources::ManagedResource, Win360::Win360Thread, Resources::LoadingResource, Win360::Win360Thread, Resources::ManagedResource, Win360::Win360Thread
- SetProgressResource() : BaseGameFeature::LoaderServer
- SetProgressText() : BaseGameFeature::LoaderServer
- SetProjMapUvOffsetAndScale() : Lighting::InternalAbstractLightEntity, Graphics::AbstractLightEntity, Lighting::InternalAbstractLightEntity, Graphics::AbstractLightEntity
- SetProjTransform() : Base::TransformDeviceBase
- SetQuadTreeSettings() : Visibility::VisibilityQuadtree
- SetQuery() : IO::URI
- SetQuitRequested() : Base::InputServerBase, App::RenderApplication, Game::GameServer, Base::InputServerBase, App::RenderApplication, Base::InputServerBase
- SetReadOnly() : IO::IoServer, OSX::OSXFSWrapper, Win360::Win360FSWrapper
- SetRecvBufSize() : Win360::Win360Socket
- SetRenderDebug() : Game::FeatureUnit, InternalGraphics::InternalGraphicsServer, Game::FeatureUnit
- SetRenderTarget() : Frame::FramePassBase, InternalGraphics::InternalView, Frame::FramePassBase
- SetResolved() : Models::VisResolveContainer< TYPE >
- SetResolveDepthTextureResourceId() : Base::RenderTargetBase
- SetResolveRect() : Base::RenderTargetBase
- SetResolveTargetCpuAccess() : Base::RenderTargetBase
- SetResolveTextureHeight() : Base::RenderTargetBase
- SetResolveTextureResourceId() : Base::RenderTargetBase
- SetResolveTextureWidth() : Base::RenderTargetBase
- SetResource() : Resources::ManagedResource
- SetResourceClass() : Resources::SimpleResourceMapper
- SetResourceCreatorClass() : Resources::PoolResourceMapper
- SetResourceId() : Resources::ManagedResource,
  CoreGraphics::MousePointer, Resources::Resource,
  Resources::ManagedResource, Graphics::ModelEntity,
  Resources::Resource, InternalGraphics::InternalModelEntity,
  Resources::Resource, Resources::ManagedResource,
  Resources::Resource
- SetResourceLoaderClass() : Resources::SimpleResourceMapper
- SetResourceMappers() : Graphics::Display
- SetResourceStreamingLevelOfDetail() : Models::ModelNode
- Set ResourceType() : Resources::ManagedResource
- SetResponseContentStream() : Http::HttpRequest
- SetResult() : Conditions::Condition
- SetReturnCode() : App::Application
- SetReuseAddr() : Win360::Win360Socket
- SetReuseTexture() : Resources::D3D9TextureStreamer
- SetRevision() : CoreGraphics::AdapterInfo
- SetRootDirectory() : Core::CoreServer
- SetRootLocation() : Http::HttpRequestHandler
- SetRootNodeOffsetMatrix() : Graphics::ModelEntity,
  InternalGraphics::InternalModelEntity
- SetRootNodePath() : Graphics::ModelEntity,
  InternalGraphics::InternalModelEntity
- SetRotate() : Models::TransformNodeInstance
- setrotate() : Math::transform44
- SetRotate() : Models::TransformNodeInstance
- SetRotatePivot() : Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode,
  Models::TransformNodeInstance, Models::TransformNode,
  Models::TransformNodeInstance
- setrotatepivot() : Math::transform44
- SetRotatePivot() : Models::TransformNodeInstance,
  Models::TransformNode, Models::TransformNodeInstance
- SetRotation() : Models::TransformNode
- setrow0() : Math::matrix44
- setrow1() : Math::matrix44
- setrow2() : Math::matrix44
- setrow3() : Math::matrix44
- SetSaveActionsFlag() : Actions::SequenceAction
- SetSaveGame() : BaseGameFeature::GameStateHandler
- SetSaver() : Resources::Resource
- SetScale() : Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode
- setscale() : Math::transform44
- SetScale() : Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode
- SetScalePivot() : Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode
- setscalepivot() : Math::transform44
- SetScalePivot() : Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode, Models::TransformNodeInstance, Models::TransformNode
- SetScheme() : IO::URI
- SetSecond() : Base::CalendarTimeBase
- SetSemantic() : Base::ShaderVariableBase
- SetSendBufSize() : Win360::Win360Socket
- SetServerAddress() : Net::StdTcpClient
- SetSetupMode() : BaseGameFeature::GameStateHandler
- SetShader() : Models::StateNode, Frame::FrameBatch, Models::StateNode, Frame::FramePassBase
- SetShaderFeatures() : Frame::FrameBatch
- SetShadowBufferUvOffsetAndScale() : Lighting::InternalAbstractLightEntity
- SetShadowIntensity() : Lighting::InternalAbstractLightEntity
- SetShadowTransform() : Lighting::InternalAbstractLightEntity
- SetSharedData() : InternalGraphics::InternalGraphicsEntity
- SetShortText() : Script::DialogTake
- SetSignalOnEnqueueEnabled() : Threading::SafeQueue<TYPE>
- SetSingleThreadMode() : Http::HttpServer
- SetSize() : IO::Stream, IO::MemoryStream, Util::FixedArray<TYPE>, Resources::ResourceDictionary::Entry, IO::Stream, Util::FixedTable<TYPE>, CoreGraphics::MousePointer,
IO::Stream

- SetSkins() : Characters::CharacterSkinList
- SetSkippedMips() : Base::TextureBase
- SetSortingMode() : Frame::FrameBatch
- SetSound() : Script::DialogTake
- SetSource() : Script::InfoLog
- SetSourceSlot() : Resources::PoolLoadingResource
- SetSpeaker() : Script::DialogTake, Script::DialogDesc
- SetStackSize() : Win360::Win360Thread, OSX::OSXThread, Win360::Win360Thread
- SetStage() : InternalGraphics::InternalView
- SetStartKeyIndex() : CoreAnimation::AnimClip
- SetStartTime() : Animation::AnimJob
- SetStatic() : CoreAnimation::AnimCurve
- SetStaticKey() : CoreAnimation::AnimCurve
- SetStatus() : Http::HttpRequest
- SetStatusCode() : Http::HttpResponseWriter
- SetStream() : IO::StreamReader, IO::StreamWriter, IO::StreamReader, Base::StreamTextureSaverBase, IO::StreamWriter, Base::StreamTextureSaverBase, IO::StreamWriter, IO::StreamReader, Messaging::MessageReader, Messaging::MessageWriter, IO::StreamReader, IO::StreamWriter, IO::StreamReader
- SetStreamByteOrder() : IO::BinaryReader, IO::BinaryWriter
- SetStreamMeshLoader() : Models::StreamModelLoader
- SetStreamSource() : Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
- SetString() : BaseGameFeature::GlobalAttrsManager, Script::ActionReader, Http::SvgPageWriter, Util::Variant,
BaseGameFeature::UserProfile, IO::XmlWriter, Game::Entity
- SetStringArray() : Util::Variant
- SetStringAttr() : Models::ModelNode
- SetSubSystemId() : CoreGraphics::AdapterInfo
- SetTargetResource() : Resources::LoadingResource
- SetTargetSlot() : Resources::PoolLoadingResource
- SetTargetState() : FSM::Transition
- SetTaskStatusUnchanged() : Script::Task
- SetTaskViewed() : Script::Task
- SetTexture() : Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable
- SetThenBlock() : Actions::IfThenElseAction
- SetTime() : CoreAnimation::AnimEvent
- SetTimeFactor() : Animation::AnimJob, FrameSync::FrameSyncTimer, BaseGameFeature::TimeManager
- SetTimeOffset() : Animation::AnimJob
- SetTimeStamp() : Script::DialogTake
- Settings() : Graphics::Display
- setTitle() : Base::GameContentServerBase, Http::HtmlPageWriter
- setTitleId() : Base::GameContentServerBase
- SetToByteOrder() : System::ByteOrder
- SetToDefault() : BaseGameFeature::UserProfile
- SetToFirstChild() : IO::BXmlReader, IO::XmlReader, IO::BXmlLoaderUtil
- SetToNextChild() : IO::BXmlLoaderUtil, IO::BXmlReader, IO::XmlReader
- SetToNode() : IO::XmlReader, IO::BXmlReader
- SetToParent() : IO::BXmlReader, IO::XmlReader, IO::BXmlLoaderUtil
- SetTrackIndex() : Animation::AnimJob
- SetTransformFromPosDirAndRange(): Graphics::PointLightEntity
- SetTransformFromPosDirRangeAndCone(): Graphics::SpotLightEntity
- SetTripleBufferingEnabled(): Base::DisplayDeviceBase, Graphics::DisplaySettings, Base::DisplayDeviceBase
- SetUnitDimensions(): Http::SvgPageWriter
- Setup(): Base::JobPortBase, Jobs::TPJobSlice, Base::MemoryIndexBufferLoaderBase, Graphics::GraphicsEntity, Base::SkinnedMeshRendererBase, Jobs::SerialJobSystem, Base::SkinnedMeshRendererBase, Jobs::TPJob, Win32::SysFunc, Particles::ParticleSystem, Base::VertexLayoutBase, Debug::DebugTimer, Models::TransformNodeInstance, Debug::DebugCounter, OSX::OSXHeap, Base::MouseRenderDeviceBase, Util::Blob, Direct3D9::D3D9ParticleRenderer, Base::JobBase, Util::QuadTree< TYPE >::Node,
Models::TransformNodeInstance,
RenderModules::RenderModule, Frame::FramePostEffect,
Characters::CharacterInstance,
Characters::CharacterVariationSet, Graphics::CameraEntity,
Base::JobPortBase, Base::MemoryVertexBufferLoaderBase,
Base::JobBase, Win32::Win32MiniDump,
Graphics::GraphicsEntity, Base::JobPortBase,
Jobs::TPJobThreadPool,
Base::MemoryIndexBufferLoaderBase, Jobs::TPJobSystem,
Util::StringBuffer, Animation::AnimSequencer,
Particles::ParticleSystemNodeInstance,
Models::ModellInstance, Jobs::TPJob,
Direct3D9::D3D9ShaderInstance, IO::ZipArchive,
Base::SkinnedMeshRendererBase, Characters::Character,
Base::ShaderVariableInstanceBase,
Models::TransformNodeInstance,
Direct3D9::D3D9ParticleRenderer, OSX::OSXMemoryPool,
Particles::EmitterMesh, Particles::EnvelopeSampleBuffer,
Direct3D9::D3D9RenderTarget,
RenderModules::RTPluginRegistry,
RenderUtil::DrawFullScreenQuad,
RenderUtil::MayaCameraUtil,
Characters::CharacterSkeletonInstance,
Base::ShaderInstanceBase, OSX::SysFunc,
Base::MemoryIndexBufferLoaderBase,
FrameSync::FrameSyncTimer,
Characters::CharacterAnimationController,
Characters::CharacterNodeInstance,
Particles::EnvelopeCurve,
Base::ShaderVariableInstanceBase, IO::ArchiveBase,
IO::ZipFileSystem, Models::ModelNodeInstance,
IO::SchemeRegistry, Win360::Win360ThreadBarrier,
Characters::CharacterSkinNodeInstance,
Jobs::TPJobSystem, Direct3D9::D3D9RenderTarget,
IO::AssignRegistry, Base::GameContentServerBase,
Base::MouseRenderDeviceBase,
CoreAnimation::AnimSampleBuffer, IO::ZipFileSystem,
Win360::Win360MemoryPool, Win360::D3D9VertexLayout,
Base::RenderTargetBase, Win360::Win360Heap,
Direct3D9::D3D9ShaderInstance,
Memory::PoolArrayAllocator, Characters::CharacterJoint,
Base::GameContentServerBase,
Characters::CharacterSkinSet, Win360::D3D9VertexLayout,
Characters::CharacterServer,
Base::MemoryVertexBufferLoaderBase,
Base::ParticleRendererBase, IO::ZipArchive,
Characters::CharacterSkeleton,
CoreAnimation::AnimKeyBuffer, Visibility::ObserverContext,
Base::MemoryVertexBufferLoaderBase

- setup_from_point_and_normal() : Math::plane
- setup_from_points() : Math::plane
- SetupAcceptedMessages() : PhysicsFeature::TriggerProperty,
  Messaging::Port, StateObjectFeature::StateProperty,
  GraphicsFeature::CameraProperty,
  GraphicsFeature::MayaCameraProperty,
  GraphicsFeature::ActorGraphicsProperty, Messaging::Port,
  GraphicsFeature::GraphicsProperty,
  GraphicsFeature::AnimationControlProperty,
  Messaging::Port, BaseGameFeature::TransformableProperty,
  Messaging::Port,
  StateObjectFeature::StateGraphicsProperty,
  PhysicsFeature::PhysicsProperty, Messaging::Port,
  Script::DialogManager,
  GraphicsFeature::ChaseCameraProperty,
  PhysicsFeature::ActorPhysicsProperty, Messaging::Port
- SetupAnimDrivenMotion() :
  Characters::CharacterAnimationController
- SetupAppFromCmdLineArgs() : App::GameApplication
- SetupAttributes() : BaseGameFeature::FactoryManager
- SetupCallbacks() : GraphicsFeature::GraphicsProperty,
  Game::Property, PhysicsFeature::MouseGripperProperty,
  Game::Property, StateObjectFeature::StateProperty,
  GraphicsFeature::MayaCameraProperty,
  GraphicsFeature::CameraProperty,
  GraphicsFeature::GraphicsProperty, Game::Property,
  PhysicsFeature::PhysicsProperty,
  GraphicsFeature::GraphicsProperty,
  PhysicsFeature::TriggerProperty,
GraphicsFeature::CameraProperty, PhysicsFeature::ActorPhysicsProperty
- SetupDefaultAttributes() : GraphicsFeature::CameraProperty, Game::Property, PhysicsFeature::TriggerProperty, GraphicsFeature::ActorGraphicsProperty, PhysicsFeature::ActorPhysicsProperty, StateObjectFeature::StateProperty, GraphicsFeature::AnimationControlProperty, GraphicsFeature::GraphicsProperty, Game::Property, GraphicsFeature::ChaseCameraProperty, Game::Property, GraphicsFeature::MayaCameraProperty, GraphicsFeature::CameraProperty, StateObjectFeature::StateGraphicsProperty, BaseGameFeature::TransformableProperty, PhysicsFeature::PhysicsProperty, PhysicsFeature::MouseGripperProperty
- SetupFromD3D9CubeTexture() : Direct3D9::D3D9Texture
- SetupFromD3D9Texture() : Direct3D9::D3D9Texture
- SetupFromD3D9VolumeTexture() : Direct3D9::D3D9Texture
- SetupFromFileInMemory() : IO::BXmlLoaderUtil
- SetupFromNax3() : CoreAnimation::StreamAnimationLoader
- SetupGameFeatures() : App::GameApplication
- SetupIndexedPrimitives() : CoreGraphics::RenderShape
- SetupJoint() : Characters::CharacterSkeleton
- SetupManagedTextureVariable() : Models::StateNode
- SetupMeshFromMemory() : CoreGraphics::MemoryMeshLoader
- SetupMultiSampleType() : Direct3D9::D3D9RenderTarget
- SetupOrthogonal() : Shared::CameraSettings
- SetupPerspectiveFov() : Shared::CameraSettings
- SetupPools() : Resources::PoolResourceMapper
- SetupPrimitives() : CoreGraphics::RenderShape
- SetupResourceFromFile() : Resources::StreamResourceLoader, Resources::D3D9TextureStreamer, CoreAnimation::StreamAnimationLoader,
Resources::D3D9TextureStreamer
- SetupRun(): Jobs::TPJobCommand
- SetupSimpleShape(): CoreGraphics::RenderShape
- SetupStateHandlers(): App::GameApplication
- SetupSync(): Jobs::TPJobCommand
- SetupTexture2DFromStream(): Resources::D3D9TextureStreamer
- SetupTextureCubeFromStream(): Resources::D3D9TextureStreamer
- SetupXAxis(): Http::SvgLineChartWriter
- SetupYAxis(): Http::SvgLineChartWriter
- SetURI(): Http::HttpRequestWriter, Resources::ResourceDictionary, IO::Stream, Http::HttpRequest, IO::Stream
- SetUsage(): Win360::D3D9StreamMeshLoader, Base::ResourceBase, Win360::D3D9StreamMeshLoader, CoreGraphics::MemoryMeshLoader, Base::ResourceBase
- SetUserAgent(): Http::HttpRequestWriter
- SetUserInfo(): IO::URI
- SetUserProfile(): BaseGameFeature::LoaderServer
- SetValid(): InternalGraphics::InternalGraphicsEntity
- SetValue(): Base::ShaderVariableInstanceBase
- SetVariationSetName(): Characters::CharacterInstance
- SetVendorId(): CoreGraphics::AdapterInfo
- SetVersion(): Base::GameContentServerBase
- SetVertexBuffer(): Base::MeshBase, CoreGraphics::MemoryMeshLoader, Base::MeshBase
- SetVertexLayout(): Direct3D9::D3D9RenderDevice, Base::VertexBufferBase, Direct3D9::D3D9RenderDevice, Base::VertexBufferBase, Base::RenderDeviceBase
- SetVerticalSyncEnabled(): Base::DisplayDeviceBase, Graphics::DisplaySettings, Base::DisplayDeviceBase
- SetViewTransform(): Base::TransformDeviceBase
- SetVisibilityContexts(): Visibility::VisibilityDevice
- SetVisibilityOnAttachments(): InternalGraphics::AttachmentServer
- SetVisible(): Graphics::GraphicsEntity, Models::ModelNodeInstance, InternalGraphics::InternalGraphicsEntity,
Graphics::GraphicsEntity,
InternalGraphics::InternalGraphicsEntity,
Models::ModelNodeInstance,
InternalGraphics::InternalGraphicsEntity,
GraphicsFeature::GraphicsProperty,
Graphics::GraphicsEntity,
InternalGraphics::InternalGraphicsEntity,
Graphics::GraphicsEntity, Models::ModelNodeInstance,
InternalGraphics::InternalGraphicsEntity,
Graphics::GraphicsEntity,
GraphicsFeature::GraphicsProperty,
Models::ModelNodeInstance,
InternalGraphics::InternalGraphicsEntity,
Models::ModelNodeInstance, Graphics::GraphicsEntity,
GraphicsFeature::GraphicsProperty,
Models::ModelNodeInstance,

- SetVisibleFrameId() : Visibility::VisibilityContext
- SetVisualizeMipMaps() : Base::RenderDeviceBase
- SetWaitTimeout() : Messaging::BlockingHandlerThread
- SetWeekday() : Base::CalendarTimeBase
- SetWeight() : Animation::AnimEventInfo
- SetWidth() : Base::RenderTargetBase,
  CoreGraphics::DisplayMode, Base::TextureBase,
  Base::RenderTargetBase
- SetWindowTitle() : Base::DisplayDeviceBase,
  Graphics::DisplaySettings, Base::DisplayDeviceBase
- SetXPos() : CoreGraphics::DisplayMode
- SetYear() : Base::CalendarTimeBase
- SetYPos() : CoreGraphics::DisplayMode
- SetZoomButton() : RenderUtil::MayaCameraUtil
- SetZoomIn() : RenderUtil::MayaCameraUtil
- SetZoomInButton() : RenderUtil::MayaCameraUtil
- SetZoomOut() : RenderUtil::MayaCameraUtil
- SetZoomOutButton() : RenderUtil::MayaCameraUtil
- ShaderBase() : Base::ShaderBase
- ShaderInstanceBase() : Base::ShaderInstanceBase
- ShaderPageHandler() : Debug::ShaderPageHandler
- ShaderServer() : CoreGraphics::ShaderServer
- ShaderServerBase() : Base::ShaderServerBase
- ShaderVariableBase: `Base::ShaderVariableBase`
- ShaderVariableInstanceBase: `Base::ShaderVariableInstanceBase`
- ShaderVariationBase: `Base::ShaderVariationBase`
- ShadowServer: `Lighting::ShadowServer`
- ShapeNode: `Models::ShapeNode`
- ShapeNodeInstance: `Models::ShapeNodeInstance`
- ShapeRenderer: `CoreGraphics::ShapeRenderer`
- ShapeRendererBase: `Base::ShapeRendererBase`
- ShowActionInfo: `Actions::Action`
- Shutdown: `Net::StdTcpClientConnection`, `Net::MessageClientConnection`, `Util::Blob`
- Signal: `Threading::SafeQueue< TYPE >`, `Win360::Win360Event`, `Threading::SafePriorityQueue< PRITYPE, TYPE >`
- SignalContinue: `Win360::Win360ThreadBarrier`
- SignalPendingResourceEvent: `Graphics::GraphicsInterface`
- SimpleResourceMapper: `Resources::SimpleResourceMapper`
- SimpleTree: `Util::SimpleTree< VALUETYPE >`
- Size: `Threading::SafePriorityQueue< PRITYPE, TYPE >`, `Threading::SafeQueue< TYPE >`, `Util::Stack< TYPE >`, `Util::SimpleTree< VALUETYPE >::Node`, `Util::Array< TYPE >`, `Util::FixedArray< TYPE >`, `Util::RingBuffer< TYPE >`
- size: `Math::bbox`
- SizeOf: `CoreGraphics::IndexType`
- Skeleton: `Characters::Character`, `Characters::CharacterInstance`
- SkinLibrary: `Characters::Character`
- SkinnedMeshRenderer: `Characters::SkinnedMeshRenderer`
- SkinnedMeshRendererBase: `Base::SkinnedMeshRendererBase`
- SkinSet: `Characters::CharacterInstance`
- SkipSegments: `PhysicsFeature::ActorPhysicsProperty`
- Sleep: `OSX::SysFunc`, `Win32::SysFunc`
- `slerp()`: `Math::quaternion`
- `SM30LightServer()`: `Lighting::SM30LightServer`
- `SM30ShadowServer()`: `Lighting::SM30ShadowServer`
- `Sort()`: `Util::Array< TYPE >`, `Util::FixedArray< TYPE >`
- `SortIfDirty()`: `Util::Dictionary< KEYTYPE, VALUETYPE >`
- `SparseTable()`: `Util::SparseTable< TYPE >`
- `sphere()`: `Math::sphere`
- `splat()`: `Math::float4`
- `splat_w()`: `Math::float4`
- `splat_x()`: `Math::float4`
- `splat_y()`: `Math::float4`
- `splat_z()`: `Math::float4`
- `SpotLightEntity()`: `Graphics::SpotLightEntity`
- `squad()`: `Math::quaternion`
- `squadsetup()`: `Math::quaternion`
- `Stack()`: `Util::Stack< TYPE >`
- `Start()`: `Actions::Action`, `Actions::SequenceAction`
- `start()`: `Math::line`
- `Start()`: `Actions::Action`, `OSX::OSXThread`, `Win360::Win360Thread`, `Game::GameServer`, `Debug::DebugTimer`, `Win360::Win360Thread`, `Actions::Action`, `Win360::Win360Timer`, `Win360::Win360Thread`
- `StartAccum()`: `Debug::DebugTimer`
- `StartAsyncEvaluation()`: `Animation::AnimSequencer`
- `StartCurrAction()`: `Actions::SequenceAction`
- `StartDialog()`: `Script::DialogManager`
- `StartEntities()`: `BaseGameFeature::EntityManager`
- `StartFixedFrameTime()`: `FrameSync::FrameSyncHandlerThread`
- `StartRenderDebug()`: `Game::FeatureUnit`
- `StartTime()`: `FrameSync::FrameSyncTimer`
- `StartTimeEffect()`: `BaseGameFeature::TimeManager`
- `StartUpdateCharacterSkeletons()`: `Characters::CharacterServer`
- `State()`: `FSM::State`
- `StateGraphicsProperty()`: `StateObjectFeature::StateGraphicsProperty`
- `StateMachine()`: `FSM::StateMachine`
- **StateNode()**: `Models::StateNode`
- **StateNodeInstance()**: `Models::StateNodeInstance`
- **StateProperty()**: `StateObjectFeature::StateProperty`
- **StateToString()**: `Script::Task`
- **StdTcpClient()**: `Net::StdTcpClient`
- **StdTcpClientConnection()**: `Net::StdTcpClientConnection`
- **StdTcpServer()**: `Net::StdTcpServer`
- **Stop()**: `Win360::Win360Thread`, `Actions::Action`, `Win360::Win360Thread`, `Actions::Action`, `Win360::Win360Thread`, `Debug::DebugTimer`, `Game::GameServer`, `Win360::Win360Timer`, `Win360::Win360Thread`, `Animation::AnimJob`, `Actions::Action`, `Actions::SequenceAction`, `Animation::AnimJob`, `Jobs::TPWorkerThread`, `Win360::Win360Thread`, `PhysicsFeature::ActorPhysicsProperty`, `OSX::OSXThread`, `Actions::Action`
- **StopAccum()**: `Debug::DebugTimer`
- **StopAllTracks()**: `Characters::CharacterAnimationController`, `Animation::AnimSequencer`
- **StopFixedFrameTime()**: `FrameSync::FrameSyncHandlerThread`
- **StopRenderDebug()**: `Game::FeatureUnit`
- **StopTime()**: `FrameSync::FrameSyncTimer`
- **StopTimeEffect()**: `BaseGameFeature::TimeManager`
- **StopTrack()**: `Characters::CharacterAnimationController`, `Animation::AnimSequencer`
- **store()**: `Math::float4`, `Math::matrix44`, `Math::quaternion`, `Math::float4`
- **storeu()**: `Math::float4`, `Math::quaternion`, `Math::float4`, `Math::matrix44`
- **StrChr()**: `Util::String`
- **StrCmp()**: `Util::String`
- **stream()**: `Math::float4`, `Math::quaternion`, `Math::float4`, `Math::matrix44`, `Math::float4`
- **Stream()**: `IO::Stream`
- **StreamingTexturePageHandler()**: `Debug::StreamingTexturePageHandler`
- **StreamModelLoader()**: `Models::StreamModelLoader`
StreamReader() : IO::StreamReader
StreamResourceLoader() : Resources::StreamResourceLoader
StreamTextureSaverBase() : Base::StreamTextureSaverBase
StreamWriter() : IO::StreamWriter
String() : Util::String
StringAtom() : Util::StringAtom
StringAtomTableBase() : Util::StringAtomTableBase
StringBuffer() : Util::StringBuffer
StringToLogLevel() : Script::InfoLog
StringToMonth() : Base::CalendarTimeBase
StringToState() : Script::Task
StringToType() : Util::Variant, Base::ShaderVariableBase, Script::DialogTake
StringToWeekday() : Base::CalendarTimeBase
Strip() : Util::String
StripFileExtension() : Util::String
StrLen() : Util::String
SubstituteChar() : Util::String
SubstituteString() : Util::String
SubstitutionManager() : Script::SubstitutionManager
SupportsDisplayMode() : Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice, Graphics::Display, Direct3D9::D3D9DisplayDevice
SvgLineChartWriter() : Http::SvgLineChartWriter
SvgPageWriter() : Http::SvgPageWriter
SvgTestPageHandler() : Debug::SvgTestPageHandler
SwitchEntity() : InternalGraphics::AttachmentServer
SwitchFocusEntities() : BaseGameFeature::FocusManager
SystemInfoBase() : Base::SystemInfoBase
SystemTimeSource() : BaseGameFeature::SystemTimeSource
SystemTimeToTick() : Win360::Win360CalendarTime, Base::CalendarTimeBase
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Enumerator
- Related Functions

- _
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
- t -

- TableRow2() : Http::HtmlPageWriter
- TalkTo() : Script::DialogManager
- Task() : Script::Task
- TcpMessageCodec() : Net::TcpMessageCodec
- Tell() : OSX::OSXFSWrapper, Win360::Win360FSWrapper
- TerminateAtIndex() : Util::String
- Test() : Threading::SafeFlag
- TestAndClearIfSet() : Threading::SafeFlag
- Text() : Http::HtmlPageWriter, Http::SvgPageWriter
- TextElement() : CoreGraphics::TextElement
- TextReader() : IO::TextReader
- TextRenderer() : CoreGraphics::TextRenderer
- TextRendererBase() : Base::TextRendererBase
- TextureBase() : Base::TextureBase
- TexturePageHandler() : Debug::TexturePageHandler
- TexturePoolMapperScheduler() : Resources::TexturePoolMapperScheduler
- TextWriter() : IO::TextWriter
- ThreadCloseHandlers() : Messaging::HandlerThreadBase
- ThreadDiscardDeferredMessages() : Messaging::HandlerThreadBase
- ThreadHandleMessages() : Messaging::HandlerThreadBase
- ThreadHandleSingleMessage() : Messaging::HandlerThreadBase
- ThreadOpenHandlers() : Messaging::HandlerThreadBase
- ThreadPageHandler() : Debug::ThreadPageHandler
- ThreadSafeDisplayEventHandler() : CoreGraphics::ThreadSafeDisplayEventHandler
- ThreadSafeRenderEventHandler() : CoreGraphics::ThreadSafeRenderEventHandler
- ThreadSignalMessageHandled() : Messaging::HandlerThreadBase
- ThreadStopRequested() : Win360::Win360Thread, OSX::OSXThread, Win360::Win360Thread
- ThreadUpdateDeferredMessages() : Messaging::HandlerThreadBase
- ThreadUpdateHandlers() : Messaging::HandlerThreadBase
- TimeManager() : BaseGameFeature::TimeManager
- TimeSource() : BaseGameFeature::TimeSource
- to_axisangle() : Math::quaternion
- to_matrix44() : Math::bbox
- TogglePerfHUD() : Debug::DebugGraphicsHandler
- ToggleSkin() : Characters::CharacterSkinSet
- ToHtml() : Http::HtmlElement
- ToHumanReadableString() : Http::HttpStatus
- Tokenize() : Util::String
- ToLower() : Util::String
- ToMediaType() : CoreGraphics::ImageFileFormat
- ToName() : Models::ModelNode
- ToString() : Frame::SortingMode, CoreAnimation::CurveType, CoreAnimation::SampleType, CoreAnimation::InfinityType, CoreGraphics::Adapter, CoreGraphics::ImageFileFormat, CoreGraphics::IndexType, CoreGraphics::PixelFormat, CoreGraphics::PrimitiveTopology, Animation::AnimJobEnqueueMode, Http::HttpMethod, CoreGraphics::AntiAliasQuality, Frame::LightingMode, Input::Key, Input::MouseButton, Http::HttpStatus, CoreGraphics::BatchType, Util::FourCC, Lighting::LightType, Script::InfoLog
- ToUpper() : Util::String
- TPJob() : Jobs::TPJob
- TPJobCommand() : Jobs::TPJobCommand
- TPJobFuncDesc() : Jobs::TPJobFuncDesc
- TPJobPort() : Jobs::TPJobPort
- TPJobSlice() : Jobs::TPJobSlice
- TPJobSystem() : Jobs::TPJobSystem
- TPJobThreadPool() : Jobs::TPJobThreadPool
- TPWorkerThread() : Jobs::TPWorkerThread
- transform() : Math::bbox, Math::matrix44
- transform44() : Math::transform44
- TransformableProperty() :
  BaseGameFeature::TransformableProperty
- transformation() : Math::matrix44
- TransformDevice() : CoreGraphics::TransformDevice
- TransformDeviceBase() : Base::TransformDeviceBase
- TransformNode() : Models::TransformNode
- TransformNodeInstance() : Models::TransformNodeInstance
- Transition() : FSM::Transition
- translate() : Math::matrix44
- TranslateKeyCode() : Win32::Win32DisplayDevice
- translation() : Math::matrix44
- transpose() : Math::matrix44
- Trigger() : Core::CoreServer, Actions::IfThenElseAction, Actions::SequenceAction, Actions::FSMAction, Actions::Action
- TriggerAction() : PhysicsFeature::TriggerProperty
- TriggerProperty() : PhysicsFeature::TriggerProperty
- Trim() : Util::String, Util::Blob
- TrimLeft() : Util::String
- TrimRight() : Util::String
- TypeToString() : Base::ShaderVariableBase, Input::InputEvent, Base::ShaderVariableBase, Util::Variant
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Alphabetical List
- Data Structures
- Class Hierarchy
- Data Fields

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Enumerator
- Related Functions

- _
- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m
- n
- o
- p
- q
- u -

- undenormalize() : **Math::quaternion**
- UnloadCanScript() : **Script::ScriptManager**
- UnloadDialog() : **Script::DialogManager**
- UnloadOnScript() : **Script::ScriptManager**
- UnloadResources() : **Characters::CharacterNode**, **Models::Model**, **Models::ShapeNode**, **Models::ModelNode**, **Models::StateNode**, **Particles::ParticleSystemNode**, **Models::ModelNode**, **Models::ShapeNode**
- UnloadStateMachines() : **Script::ScriptManager**
- UnloadSubTasks() : **Script::Task**
- Unlock() : **Resources::Resource**
- UnlockSurfaces() : **Resources::D3D9TextureStreamer**
- UnlockTask() : **Script::Task**
- Unmap() : **Win360::D3D9IndexBuffer**, **CoreAnimation::AnimKeyBuffer**, **IO::ZipFileStream**,
UnmapCubeFace() : Direct3D9::D3D9Texture, Base::TextureBase, Direct3D9::D3D9Texture
UnmapSampleCounts() : CoreAnimation::AnimSampleBuffer
UnmapSamples() : CoreAnimation::AnimSampleBuffer
UnmountArchive() : IO::IoServer
UnmountStandardArchives() : IO::IoServer
unpack_w() : Math::float4
unpack_x() : Math::float4
unpack_y() : Math::float4
unpack_z() : Math::float4
UnregisterAnimEventHandler() : Animation::AnimEventServer
UnregisterDebugCounter() : Debug::DebugServer
UnregisterDebugTimer() : Debug::DebugServer
UnregisterEntity() : Visibility::VisibilityChecker
UnregisterRenderModule() : Graphics::GraphicsServer
UnregisterRTPlugin() : RenderModules::RTPluginRegistry
UnregisterUnmanagedResource() : Resources::ResourceManager
UnregisterUriScheme() : IO::SchemeRegistry
upcast() : Ptr< TYPE >
Update() : IO::ConsoleHandler, Jobs::JobDataDesc, RenderUtil::MayaCameraUtil, Jobs::JobUniformDesc, BaseGameFeature::TimeManager, IO::Console, IO::ConsoleHandler, Resources::ResourceManager, IO::ConsoleHandler, GraphicsFeature::LightFlickerUtil, IO::ConsoleHandler, FSM::StateMachine
UpdateAudioListenerPosition() : GraphicsFeature::CameraProperty
UpdateBoundingBox() : Models::Model
UpdateButtonState() : XInput::XInputGamePad
UpdateCamera() : GraphicsFeature::ChaseCameraProperty
UpdateCameraLinks() : InternalGraphics::InternalStage
UpdateCharacterSkins() : Characters::CharacterServer
UpdateClipStatus() : InternalGraphics::InternalGraphicsEntity
- UpdateDialogData(): `Script::Dialog`
- UpdateGlobalBoundingBox(): `InternalGraphics::InternalGraphicsEntity`
- UpdateLightLinks(): `InternalGraphics::InternalStage`
- UpdateManagedTextureVariables(): `Models::StateNode`
- UpdateModelNodeInstanceShaderVariable(): `Graphics::ModelEntity`
- UpdateModelNodeInstanceVisibility(): `Graphics::ModelEntity`
- UpdatePointers(): `Base::MouseRenderDeviceBase, Graphics::MouseRenderer`
- UpdateProgressDisplay(): `BaseGameFeature::LoaderServer`
- UpdateRenderStats(): `Resources::ManagedResource, Models::ModelInstance, Resources::ManagedResource`
- UpdateShadowBuffers(): `Lighting::SM30ShadowServer`
- UpdateShadowTransforms(): `Lighting::InternalAbstractLightEntity`
- UpdateSoftwareSkinnedMeshes(): `Base::SkinnedMeshRendererBase`
- UpdateThumbAxis(): `XInput::XInputGamePad`
- UpdateTimePolling(): `FrameSync::FrameSyncTimer`
- UpdateTimes(): `Animation::AnimJob`
- UpdateTransform(): `GraphicsFeature::GraphicsProperty`
- UpdateTransforms(): `StateObjectFeature::StateGraphicsProperty`
- UpdateTriggerAxis(): `XInput::XInputGamePad`
- UpdateTriggeredEntities(): `BaseGameFeature::EntityManager`
- UpdateVertexStreams(): `Direct3D9::D3D9ParticleSystemInstance`
- UpdateViewMatrix(): `Shared::CameraSettings`
- UpdateVisibilityContext(): `Visibility::VisibilityChecker, Visibility::VisibilityQuadtree, Visibility::VisibilitySystemBase, Visibility::VisibilityClusterSystem, Visibility::VisibilityBoxSystem`
- UpdateVisibilityLinks() : `InternalGraphics::InternalView`
- upvec() : `Math::vector`
- URI() : `IO::URI`
- UserInfo() : `IO::URI`
- UserProfile() : `BaseGameFeature::UserProfile`
- UTF8ToWide() : `Win32::Win32StringConverter`
- v -

- Validate() : Threading::ObjectRef
- ValidateDialog() : Script::DialogManager
- ValidateModelInstance() : 
  InternalGraphics::InternalModelEntity
- Value() : Util::KeyValuePair<KEYTYPE, VALUETYPE>, Util::SimpleTree<VALUETYPE>::Node, Util::KeyValuePair<KEYTYPE, VALUETYPE>, Util::SimpleTree<VALUETYPE>::Node, Util::StringAtom
- ValueAtIndex() : Util::Dictionary<KEYTYPE, VALUETYPE>
- Values() : BaseGameFeature::CategoryManager::Entry
- ValuesAs() : Util::Dictionary<KEYTYPE, VALUETYPE>
- ValuesAsArray() : Util::Dictionary<KEYTYPE, VALUETYPE>
- Variant() : Util::Variant
- VariationLibrary() : Characters::Character
- vec() : Math::line
- vector() : Math::vector
- Version() : Script::ActionReader
- VertexBufferBase() : Base::VertexBufferBase
- VertexComponent() : CoreGraphics::VertexComponent
- VertexLayoutBase() : Base::VertexLayoutBase
- VertexLayoutServerBase() : Base::VertexLayoutServerBase
- View() : Graphics::View
- ViewerApplication() : App::ViewerApplication
- VisibilityBoxSystem() : Visibility::VisibilityBoxSystem
- VisibilityChecker() : Visibility::VisibilityChecker
- VisibilityClusterSystem() : Visibility::VisibilityClusterSystem
- VisibilityContainer() : Visibility::VisibilityContainer
- VisibilityContext() : Visibility::VisibilityContext
- VisibilityQuadtree() : Visibility::VisibilityQuadtree
- VisibilityQuery() : Visibility::VisibilityQuery
- VisibilitySystemBase() : Visibility::VisibilitySystemBase
- VisResolveContainer() : Models::VisResolveContainer< TYPE >
- VisResolver() : Models::VisResolver
- w -

- Math::float4, Math::quaternion
- Wait() : Threading::SafePriorityQueue< PRITYPE, TYPE >, Threading::SafeQueue< TYPE >, Messaging::AsyncPort, Win360::Win360Event, Win360::Win360ThreadBarrier, Messaging::AsyncPort
- WaitDone() : Jobs::TPJobPort, Base::JobPortBase, Jobs::SerialJobPort
- WaitForFinished() : Visibility::VisibilityQuery
- WaitForHandlersOpened() : Messaging::HandlerThreadBase
- WaitForMessage() : Messaging::HandlerThreadBase, FrameSync::FrameSyncHandlerThread, Messaging::HandlerThreadBase
- WaitForPendingResources() : Graphics::GraphicsHandler, Graphics::GraphicsInterface, Resources::ResourceManager
- WaitTimeout() : Win360::Win360Event, Threading::SafeQueue< TYPE >, Win360::Win360Event
- WaitUpdateDone() : Characters::CharacterInstance
- Warning() : IO::HistoryConsoleHandler, IO::LogFileConsoleHandler, IO::ConsoleHandler, OSX::OSXConsoleHandler, Win32::Win32ConsoleHandler, IO::Console
- WeakPtr() : WeakPtr< TYPE >
- WeekdayToString() : Base::CalendarTimeBase
- WheelBackward() : Base::MouseBase
- WheelForward() : Base::MouseBase
- WideToUTF8() : Win32::Win32StringConverter
- Width() : Util::FixedTable< TYPE >
- width() : Math::rectangle< TYPE >
- Win32ConsoleHandler() : Win32::Win32ConsoleHandler
- Win32DisplayDevice() : Win32::Win32DisplayDevice
- Win32Guid() : Win32::Win32Guid
- Win32InputServer() : Win32::Win32InputServer
- Win32SkinnedMeshRenderer() : Win32::Win32SkinnedMeshRenderer
- Win32SystemInfo() : Win32::Win32SystemInfo
- Win360CriticalSection() : Win360::Win360CriticalSection
- Win360Event() : Win360::Win360Event
- Win360FileTime() : Win360::Win360FileTime
- Win360Heap() : Win360::Win360Heap
- Win360IpAddress() : Win360::Win360IpAddress
- Win360MemoryPool() : Win360::Win360MemoryPool
- Win360Socket() : Win360::Win360Socket
- Win360Thread() : Win360::Win360Thread
- Win360Thread Barrier() : Win360::Win360ThreadBarrier
- Win360Timer() : Win360::Win360Timer
- WinProc() : Win32::Win32DisplayDevice
- Write() : IO::FileStream, Actions::SequenceAction, IO::Stream, Conditions::And, Actions::IfThenElseAction, IO::Stream, IO::MemoryStream, OSX::OSXFSWrapper, Conditions::Condition, IO::TextWriter, IO::Stream, Win360::Win360FSWrapper, Conditions::Or, Actions::Action, Actions::ActionList, Conditions::Condition, Conditions::Not, Actions::Action, Net::DebugPacket, IO::Stream
- WriteBlob() : IO::BinaryWriter
- WriteBool() : IO::BinaryWriter
- WriteChar() : IO::TextWriter, IO::BinaryWriter
- WriteComment() : IO::XmlWriter
- WriteContent() : Http::SvgPageWriter, IO::XmlWriter, Http::SvgPageWriter
- WriteDouble() : IO::BinaryWriter
- WriteFloat() : IO::BinaryWriter
- WriteFloat2() : IO::BinaryWriter
- WriteFloat4() : **IO::BinaryWriter**
- WriteFloatAsNormalizedUByte2() : **IO::BinaryWriter**
- WriteFloatAsUnsignedNormalizedUByte2() : **IO::BinaryWriter**
- WriteFormatted() : **IO::TextWriter**
- WriteGuid() : **IO::BinaryWriter**
- WriteInt() : **IO::BinaryWriter**
- WriteLine() : **IO::TextWriter**
- WriteLines() : **IO::TextWriter**
- WriteMatrix44() : **IO::BinaryWriter**
- WriteMessage() : **Messaging::MessageWriter**
- WriteMiniDump() : **Win32::Win32MiniDump**
- WritePoint() : **IO::BinaryWriter**
- WritePoolsToXML() : **Resources::PoolResourceMapper**
- WriteRaw() : **Net::DebugPacket**
- WriteRawData() : **IO::BinaryWriter**
- WriteRequestHeader() : **Http::HttpRequestWriter**
- WriteResponse() : **Http::HttpResponseWriter**
- WriteShort() : **IO::BinaryWriter**
- WriteString() : **IO::BinaryWriter**, **System::Win32Registry**, **IO::TextWriter**
- WriteTexturePoolToXML() : **Resources::PoolResourceMapper**
- WriteUChar() : **IO::BinaryWriter**
- WriteUInt() : **IO::BinaryWriter**
- WriteUShort() : **IO::BinaryWriter**
- WriteVector() : **IO::BinaryWriter**
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_ 
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
- x -

- x() : Math::float2, Math::float4, Math::float2, Math::float4, Math::quat
- XInputGamePad() : XInput::XInputGamePad
- XmlReader() : IO::XmlReader
- XmlWriter() : IO::XmlWriter
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_ 
 a 
b 
c 
d 
e 
f 
g 
h 
i 
j 
k 
l 
m 
n 
o 
p 
q
- y -

- y() : Math::float2, Math::float4, Math::quaternion, Math::float4, Math::quaternion, Math::float4
- YieldThread() : Win360::Win360Thread, OSX::OSXThread, Win360::Win360Thread

The Nebula Device 3 documentation generated by doxygen at Fri Mar 26 15:21:40 2010
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Enumerator
Related Functions

_ 
a 
b 
c 
d 
e 
f 
g 
h 
i 
j 
k 
l 
m 
n 
o 
p 
q
- Z -

- z() : Math::float4, Math::quaternion, Math::float4, Math::quaternion, Math::float4
- ZipArchive() : IO::ZipArchive
- ZipDirEntry() : IO::ZipDirEntry
- ZipFileEntry() : IO::ZipFileEntry
- ZipFileStream() : IO::ZipFileStream
- ZipFileSystem() : IO::ZipFileSystem
AbstractLightEntity() : Graphics::AbstractLightEntity
~Action() : Actions::Action
~ActionReader() : Script::ActionReader
~ActorPhysicsProperty() :
  PhysicsFeature::ActorPhysicsProperty
~AnimEventHandlerBase() :
  Animation::AnimEventHandlerBase
~AnimEventManager() :
  BaseGameFeature::AnimEventManager
~AnimEventServer() : Animation::AnimEventServer
~AnimJob() : Animation::AnimJob
~AnimKeyBuffer() : CoreAnimation::AnimKeyBuffer
~AnimResource() : CoreAnimation::AnimResource
~AnimSampleBuffer() : CoreAnimation::AnimSampleBuffer
~AnimSequencer() : Animation::AnimSequencer
~Application() : App::Application
~ArchiveBase() : IO::ArchiveBase
~ArchiveFileSystem() : IO::ArchiveFileSystem
~Array() : Util::Array< TYPE >
~AssignRegistry() : IO::AssignRegistry
~AsyncPort() : Messaging::AsyncPort
~AttachmentManager() :
  GraphicsFeature::AttachmentManager
~AttachmentServer() : InternalGraphics::AttachmentServer
- ~BinaryReader() : IO::BinaryReader
- ~BinaryWriter() : IO::BinaryWriter
- ~Blob() : Util::Blob
- ~BXmlLoaderUtil() : IO::BXmlLoaderUtil
- ~BXmlReader() : IO::BXmlReader
- ~CameraEntity() : Graphics::CameraEntity
- ~CameraProperty() : GraphicsFeature::CameraProperty
- ~CategoryManager() : BaseGameFeature::CategoryManager
- ~Character() : Characters::Character
- ~CharacterAnimationController() :
  Characters::CharacterAnimationController
- ~CharacterInstance() : Characters::CharacterInstance
- ~CharacterJoint() : Characters::CharacterJoint
- ~CharacterNode() : Characters::CharacterNode
- ~CharacterNodeInstance() :
  Characters::CharacterNodeInstance
- ~CharacterServer() : Characters::CharacterServer
- ~CharacterSkeleton() : Characters::CharacterSkeleton
- ~CharacterSkeletonInstance() :
  Characters::CharacterSkeletonInstance
- ~CharacterSkin() : Characters::CharacterSkin
- ~CharacterSkinLibrary() : Characters::CharacterSkinLibrary
- ~CharacterSkinList() : Characters::CharacterSkinList
- ~CharacterSkinNode() : Characters::CharacterSkinNode
- ~CharacterSkinNodeInstance() :
  Characters::CharacterSkinNodeInstance
- ~CharacterSkinSet() : Characters::CharacterSkinSet
- ~CharacterVariationSet() : Characters::CharacterVariationSet
- ~ChaseCameraProperty() :
  GraphicsFeature::ChaseCameraProperty
- ~Console() : IO::Console
- ~ConsoleApplication() : App::ConsoleApplication
- ~ConsoleHandler() : IO::ConsoleHandler
- ~ConsolePageHandler() : Debug::ConsolePageHandler
- ~CoreServer() : Core::CoreServer
- ~CreateEntityCommand() : Commands::CreateEntityCommand
- ~D3D9DisplayDevice() : Direct3D9::D3D9DisplayDevice
- ~D3D9IndexBuffer() : Win360::D3D9IndexBuffer
- ~D3D9ParticleRenderer() : Direct3D9::D3D9ParticleRenderer
~D3D9ParticleSystemInstance() : Direct3D9::D3D9ParticleSystemInstance
~D3D9RenderDevice() : Direct3D9::D3D9RenderDevice
~D3D9RenderTarget() : Direct3D9::D3D9RenderTarget
~D3D9Shader() : Direct3D9::D3D9Shader
~D3D9ShaderInstance() : Direct3D9::D3D9ShaderInstance
~D3D9ShaderServer() : Direct3D9::D3D9ShaderServer
~D3D9ShaderVariable() : Direct3D9::D3D9ShaderVariable
~D3D9ShaderVariation() : Direct3D9::D3D9ShaderVariation
~D3D9ShapeRenderer() : Win360::D3D9ShapeRenderer
~D3D9TextRenderer() : Direct3D9::D3D9TextRenderer
~D3D9Texture() : Direct3D9::D3D9Texture
~D3D9TransformDevice() : Win360::D3D9TransformDevice
~D3D9VertexBuffer() : Win360::D3D9VertexBuffer
~D3D9VertexLayout() : Win360::D3D9VertexLayout
~DebugCounter() : Debug::DebugCounter
~DebugGraphicsHandler() : Debug::DebugGraphicsHandler
~DebugHandler() : Debug::DebugHandler
~DebugInterface() : Debug::DebugInterface
~DebugPacket() : Net::DebugPacket
~DebugServer() : Debug::DebugServer
~DebugShapeRenderer() : Debug::DebugShapeRenderer
~DebugTextRenderer() : Debug::DebugTextRenderer
~DebugTimer() : Debug::DebugTimer
~Dialog() : Script::Dialog
~DialogManager() : Script::DialogManager
~DialogTake() : Script::DialogTake
~Display() : Graphics::Display
~DisplayDevice() : CoreGraphics::DisplayDevice
~DisplayDeviceBase() : Base::DisplayDeviceBase
~DisplayEventHandler() : CoreGraphics::DisplayEventHandler
~DrawFullScreenQuad() : RenderUtil::DrawFullScreenQuad
~EmitterMesh() : Particles::EmitterMesh
~Entity() : Game::Entity
~EntityLoaderBase() : BaseGameFeature::EntityLoaderBase
~EntityManager() : BaseGameFeature::EntityManager
~EnvelopeSampleBuffer() : Particles::EnvelopeSampleBuffer
~EnvEntityManager() : BaseGameFeature::EnvEntityManager
~EnvironmentCollideProperty() :
PhysicsFeature::EnvironmentCollideProperty
- ~EnvQueryManager() : BaseGameFeature::EnvQueryManager
- ~ExcelXmlReader() : IO::ExcelXmlReader
- ~ExitHandler() : Core::ExitHandler
- ~Extrapolator() : Math::Extrapolator< TYPE >
- ~FactoryManager() : BaseGameFeature::FactoryManager
- ~FeatureUnit() : Game::FeatureUnit
- ~FileStream() : IO::FileStream
- ~FixedArray() : Util::FixedArray< TYPE >
- ~FixedTable() : Util::FixedTable< TYPE >
- ~FocusManager() : BaseGameFeature::FocusManager
- ~FrameBatch() : Frame::FrameBatch
- ~FramePass() : Frame::FramePass
- ~FramePassBase() : Frame::FramePassBase
- ~FramePostEffect() : Frame::FramePostEffect
- ~FrameServer() : Frame::FrameServer
- ~FrameShader() : Frame::FrameShader
- ~FrameSyncHandlerThread() :
  FrameSync::FrameSyncHandlerThread
- ~FrameSyncSharedData() :
  FrameSync::FrameSyncSharedData
- ~FrameSyncTimer() : FrameSync::FrameSyncTimer
- ~GameApplication() : App::GameApplication
- ~GameContentServer() : IO::GameContentServer
- ~GameContentServerBase() : Base::GameContentServerBase
- ~GamePadBase() : Base::GamePadBase
- ~GameServer() : Game::GameServer
- ~GameStateHandler() : BaseGameFeature::GameStateHandler
- ~GlobalAttrsManager() :
  BaseGameFeature::GlobalAttrsManager
- ~GlobalStringAtomTable() : Util::GlobalStringAtomTable
- ~GraphicsEntity() : Graphics::GraphicsEntity
- ~GraphicsHandler() : Graphics::GraphicsHandler
- ~GraphicsInterface() : Graphics::GraphicsInterface
- ~GraphicsProperty() : GraphicsFeature::GraphicsProperty
- ~GraphicsServer() : Graphics::GraphicsServer
- ~Handler() : Messaging::Handler
- ~HtmlPageWriter() : Http::HtmlPageWriter
- ~HttpInterface() : Http::HttpInterface
~HttpMessageHandler() : Http::HttpMessageHandler
~HttpRequest() : Http::HttpRequest
~HttpRequestHandler() : Http::HttpRequestHandler
~HttpServer() : Http::HttpServer
~HttpServerProxy() : Http::HttpServerProxy
~IndexBufferBase() : Base::IndexBufferBase
~InfoLog() : Script::InfoLog
~InputHandler() : Input::InputHandler
~InputServer() : Input::InputServer
~InputServerBase() : Base::InputServerBase
~InputTimeSource() : BaseGameFeature::InputTimeSource
~InterfaceBase() : Interface::InterfaceBase
~InternalAbstractLightEntity() : 
  Lighting::AbstractLightEntity
~InternalCameraEntity() : 
  InternalGraphics::InternalCameraEntity
~InternalGraphicsEntity() : 
  InternalGraphics::InternalGraphicsEntity
~InternalGraphicsServer() : 
  InternalGraphics::InternalGraphicsServer
~InternalModelEntity() : InternalGraphics::InternalModelEntity
~InternalStage() : InternalGraphics::InternalStage
~InternalView() : InternalGraphics::InternalView
~IoInterfaceHandler() : IO::IoInterfaceHandler
~IoServer() : IO::IoServer
~JobBase() : Base::JobBase
~JobPortBase() : Base::JobPortBase
~JobSystem() : Jobs::JobSystem
~KeyboardBase() : Base::KeyboardBase
~LightPrePassServer() : Lighting::LightPrePassServer
~LightServer() : Lighting::LightServer
~LightServerBase() : Lighting::LightServerBase
~List() : Util::List< TYPE >
~LoaderServer() : BaseGameFeature::LoaderServer
~LoadingResource() : Resources::LoadingResource
~LocalStringAtomTable() : Util::LocalStringAtomTable
~LogFileConsoleHandler() : IO::LogFileConsoleHandler
~ManagedResource() : Resources::ManagedResource
~Manager() : Game::Manager
~MayaCameraProperty() : 
  GraphicsFeature::MayaCameraProperty
~MemoryStream() : IO::MemoryStream
~MeshBase() : Base::MeshBase
~MessageClient() : Net::MessageClient
~Model() : Models::Model
~ModelEntity() : Graphics::ModelEntity
~ModelInstance() : Models::ModelInstance
~ModelNode() : Models::ModelNode
~ModelNodeInstance() : Models::ModelNodeInstance
~ModelServer() : Models::ModelServer
~MouseBase() : Base::MouseBase
~MouseGripperProperty() :
  PhysicsFeature::MouseGripperProperty
~MouseRenderDevice() : CoreGraphics::MouseRenderDevice
~MouseRenderDeviceBase() : Base::MouseRenderDeviceBase
~MouseRenderer() : Graphics::MouseRenderer
~MultipleRenderTargetBase() :
  Base::MultipleRenderTargetBase
~Node() : Util::SimpleTree< VALUETYPE >::Node ,
  Util::QuadTree< TYPE >::Node
~ObjectRef() : Threading::ObjectRef
~ObserverContext() : Visibility::ObserverContext
~OSXCriticalSection() : OSX::OSXCriticalSection
~OSXHeap() : OSX::OSXHeap
~OSXMemoryPool() : OSX::OSXMemoryPool
~OSXThread() : OSX::OSXThread
~OSXThreadLocalPtr() : OSX::OSXThreadLocalPtr< TYPE >
~ParticleRenderer() : Particles::ParticleRenderer
~ParticleRendererBase() : Base::ParticleRendererBase
~ParticleServer() : Particles::ParticleServer
~ParticleSystem() : Particles::ParticleSystem
~ParticleSystemNode() : Particles::ParticleSystemNode
~ParticleSystemNodeInstance() :
  Particles::ParticleSystemNodeInstance
~PhysicsProperty() : PhysicsFeature::PhysicsProperty
~PlayClipJob() : Animation::PlayClipJob
~PoolArrayAllocator() : Memory::PoolArrayAllocator
~PoolLoadingResource() : Resources::PoolLoadingResource
- ~PoolResourceMapper() : Resources::PoolResourceMapper
- ~Property() : Game::Property
- ~Ptr() : Ptr< TYPE >
- ~RefCounted() : Core::RefCounted
- ~RenderApplication() : App::RenderApplication
- ~RenderDevice() : CoreGraphics::RenderDevice
- ~RenderDeviceBase() : Base::RenderDeviceBase
- ~RenderEventHandler() : CoreGraphics::RenderEventHandler
- ~RenderModule() : RenderModules::RenderModule
- ~RenderTargetBase() : Base::RenderTargetBase
- ~Resource() : Resources::Resource
- ~ResourceBase() : Base::ResourceBase
- ~ResourceDictionary() : Resources::ResourceDictionary
- ~ResourceLoader() : Resources::ResourceLoader
- ~ResourceManager() : Resources::ResourceManager
- ~ResourceMapper() : Resources::ResourceMapper
- ~ResourceSaver() : Resources::ResourceSaver
- ~RingBuffer() : Util::RingBuffer< TYPE >
- ~RTPlugin() : RenderModules::RTPlugin
- ~RTPluginRegistry() : RenderModules::RTPluginRegistry
- ~SchemeRegistry() : IO::SchemeRegistry
- ~ScriptFeatureUnit() : ScriptFeature::ScriptFeatureUnit
- ~ScriptManager() : Script::ScriptManager
- ~SequenceAction() : Actions::SequenceAction
- ~SerialJob() : Jobs::SerialJob
- ~SerialJobPort() : Jobs::SerialJobPort
- ~SerialJobSystem() : Jobs::SerialJobSystem
- ~ShaderBase() : Base::ShaderBase
- ~ShaderInstanceBase() : Base::ShaderInstanceBase
- ~ShaderServer() : CoreGraphics::ShaderServer
- ~ShaderServerBase() : Base::ShaderServerBase
- ~ShaderVariableBase() : Base::ShaderVariableBase
- ~ShaderVariableInstanceBase() :
  Base::ShaderVariableInstanceBase
- ~ShaderVariationBase() : Base::ShaderVariationBase
- ~ShadowServer() : Lighting::ShadowServer
- ~ShapeNode() : Models::ShapeNode
- ~ShapeNodeInstance() : Models::ShapeNodeInstance
- ~ShapeRenderer() : CoreGraphics::ShapeRenderer
- `~ShapeRendererBase()` : `Base::ShapeRendererBase`
- `~SimpleResourceMapper()` : `Resources::SimpleResourceMapper`
- `~SkinnedMeshRenderer()` : `Characters::SkinnedMeshRenderer`
- `~SkinnedMeshRendererBase()` : `Base::SkinnedMeshRendererBase`
- `~SM30LightServer()` : `Lighting::SM30LightServer`
- `~SM30ShadowServer()` : `Lighting::SM30ShadowServer`
- `~State()` : `FSM::State`
- `~StateGraphicsProperty()` : `StateObjectFeature::StateGraphicsProperty`
- `~StateMachine()` : `FSM::StateMachine`
- `~StateNode()` : `Models::StateNode`
- `~StateNodeInstance()` : `Models::StateNodeInstance`
- `~StdTcpClient()` : `Net::StdTcpClient`
- `~StdTcpClientConnection()` : `Net::StdTcpClientConnection`
- `~StdTcpServer()` : `Net::StdTcpServer`
- `~Stream()` : `IO::Stream`
- `~StreamModelLoader()` : `Models::StreamModelLoader`
- `~StreamReader()` : `IO::StreamReader`
- `~StreamResourceLoader()` : `Resources::StreamResourceLoader`
- `~StreamTextureSaverBase()` : `Base::StreamTextureSaverBase`
- `~StreamWriter()` : `IO::StreamWriter`
- `~String()` : `Util::String`
- `~StringAtomTableBase()` : `Util::StringAtomTableBase`
- `~StringBuffer()` : `Util::StringBuffer`
- `~SubstitutionManager()` : `Script::SubstitutionManager`
- `~SvgLineChartWriter()` : `Http::SvgLineChartWriter`
- `~SvgPageWriter()` : `Http::SvgPageWriter`
- `~SystemTimeSource()` : `BaseGameFeature::SystemTimeSource`
- `~Task()` : `Script::Task`
- `~TcpMessageCodec()` : `Net::TcpMessageCodec`
- `~TextReader()` : `IO::TextReader`
- `~TextRenderer()` : `CoreGraphics::TextRenderer`
- `~TextRendererBase()` : `Base::TextRendererBase`
- `~TextureBase()` : `Base::TextureBase`
- `~TexturePoolMapperScheduler()` :
Resources::TexturePoolMapperScheduler
- ~TextWriter() : IO::TextWriter
- ~ThreadSafeDisplayEventHandler() :
  CoreGraphics::ThreadSafeDisplayEventHandler
- ~ThreadSafeRenderEventHandler() :
  CoreGraphics::ThreadSafeRenderEventHandler
- ~TimeManager() : BaseGameFeature::TimeManager
- ~TimeSource() : BaseGameFeature::TimeSource
- ~TPJob() : Jobs::TPJob
- ~TPJobPort() : Jobs::TPJobPort
- ~TPJobSlice() : Jobs::TPJobSlice
- ~TPJobSystem() : Jobs::TPJobSystem
- ~TPJobThreadPool() : Jobs::TPJobThreadPool
- ~TPWorkerThread() : Jobs::TPWorkerThread
- ~TransformableProperty() :
  BaseGameFeature::TransformableProperty
- ~TransformDevice() : CoreGraphics::TransformDevice
- ~TransformDeviceBase() : Base::TransformDeviceBase
- ~TransformNode() : Models::TransformNode
- ~TransformNodeInstance() : Models::TransformNodeInstance
- ~Transition() : FSM::Transition
- ~TriggerProperty() : PhysicsFeature::TriggerProperty
- ~UserProfile() : BaseGameFeature::UserProfile
- ~Variant() : Util::Variant
- ~VertexBufferBase() : Base::VertexBufferBase
- ~VertexLayoutBase() : Base::VertexLayoutBase
- ~VertexLayoutServerBase() : Base::VertexLayoutServerBase
- ~View() : Graphics::View
- ~ViewerApplication() : App::ViewerApplication
- ~VisibilityBoxSystem() : Visibility::VisibilityBoxSystem
- ~VisibilityChecker() : Visibility::VisibilityChecker
- ~VisibilityClusterSystem() : Visibility::VisibilityClusterSystem
- ~VisibilityContainer() : Visibility::VisibilityContainer
- ~VisibilityContext() : Visibility::VisibilityContext
- ~VisibilityQuadtree() : Visibility::VisibilityQuadtree
- ~VisibilityQuery() : Visibility::VisibilityQuery
- ~VisibilitySystemBase() : Visibility::VisibilitySystemBase
- ~VisResolver() : Models::VisResolver
- ~WeakPtr() : WeakPtr<TYPE>
- ~Win32DisplayDevice() : Win32::Win32DisplayDevice
- ~Win32InputServer() : Win32::Win32InputServer
- ~Win32SkinnedMeshRenderer() :
  Win32::Win32SkinnedMeshRenderer
- ~Win360CriticalSection() : Win360::Win360CriticalSection
- ~Win360Event() : Win360::Win360Event
- ~Win360Heap() : Win360::Win360Heap
- ~Win360MemoryPool() : Win360::Win360MemoryPool
- ~Win360Socket() : Win360::Win360Socket
- ~Win360Thread() : Win360::Win360Thread
- ~Win360ThreadBarrier() : Win360::Win360ThreadBarrier
- ~XInputGamePad() : XInput::XInputGamePad
- ~XmlReader() : IO::XmlReader
- ~XmlWriter() : IO::XmlWriter
- ~ZipArchive() : IO::ZipArchive
- ~ZipFileEntry() : IO::ZipFileEntry
- ~ZipFileStream() : IO::ZipFileStream
- ~ZipFileSystem() : IO::ZipFileSystem