The Nebula Device 3 Data Structures

Here are the data structures with brief descriptions:

<table>
<thead>
<tr>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>AngularPFeedbackLoop</td>
</tr>
<tr>
<td>App::Application</td>
</tr>
<tr>
<td>App::AsyncRenderApplication</td>
</tr>
<tr>
<td>App::AsyncViewerApplication</td>
</tr>
<tr>
<td>App::ConsoleApplication</td>
</tr>
<tr>
<td>App::GameApplication</td>
</tr>
<tr>
<td>App::GameStateHandler</td>
</tr>
<tr>
<td>App::RenderApplication</td>
</tr>
<tr>
<td>App::ViewerApplication</td>
</tr>
<tr>
<td>Application::StateHandler</td>
</tr>
<tr>
<td>AsyncGraphics::AsyncGraphicsHandler</td>
</tr>
<tr>
<td>AsyncGraphics::AsyncGraphicsInterface</td>
</tr>
<tr>
<td>AsyncGraphics::CameraEntityProxy</td>
</tr>
<tr>
<td>AsyncGraphics::DisplayProxy</td>
</tr>
<tr>
<td>AsyncGraphics::GraphicsEntityProxy</td>
</tr>
<tr>
<td>AsyncGraphics::GraphicsServerProxy</td>
</tr>
<tr>
<td>AsyncGraphics::Handle</td>
</tr>
<tr>
<td>AsyncGraphics::ViewProxy</td>
</tr>
<tr>
<td>AsyncHttp::AsyncHttpHandler</td>
</tr>
<tr>
<td>AsyncHttp::AsyncHttpInterface</td>
</tr>
<tr>
<td>AsyncHttp::HttpServerProxy</td>
</tr>
<tr>
<td>Attr::AccessMode</td>
</tr>
<tr>
<td>Attr::Attribute</td>
</tr>
<tr>
<td>Attr::AttributeContainer</td>
</tr>
<tr>
<td>Attr::AttributeDefinition&lt; VALUETYPE, TYPE &gt;</td>
</tr>
<tr>
<td>Attr::AttributeDefinitionBase</td>
</tr>
<tr>
<td>Attr::AttributeTable</td>
</tr>
<tr>
<td>Attr::AttrId</td>
</tr>
<tr>
<td>Attr::BlobAttrId</td>
</tr>
<tr>
<td>Attr::BoolAttrId</td>
</tr>
<tr>
<td>Attr::Float4AttrId</td>
</tr>
<tr>
<td>Attr::FloatAttrId</td>
</tr>
<tr>
<td>Attr::GuidAttrId</td>
</tr>
<tr>
<td>Attr::IntAttrId</td>
</tr>
<tr>
<td>Base::CalendarTimeBase</td>
</tr>
<tr>
<td>Base::D3D9TransformDevice</td>
</tr>
<tr>
<td>Base::DisplayDeviceBase</td>
</tr>
<tr>
<td>Base::GamePadBase</td>
</tr>
<tr>
<td>Base::IndexBufferBase</td>
</tr>
<tr>
<td>Base::InputServerBase</td>
</tr>
<tr>
<td>Base::KeyboardBase</td>
</tr>
<tr>
<td>Base::MemoryIndexBufferLoaderBase</td>
</tr>
<tr>
<td>Base::MemoryVertexBufferLoaderBase</td>
</tr>
<tr>
<td>Base::MeshBase</td>
</tr>
<tr>
<td>Base::MouseBase</td>
</tr>
<tr>
<td>Base::PreShader</td>
</tr>
<tr>
<td>Base::RenderDeviceBase</td>
</tr>
<tr>
<td>Base::RenderTargetBase</td>
</tr>
<tr>
<td>Base::ResourceBase</td>
</tr>
<tr>
<td>Base::ShaderBase</td>
</tr>
<tr>
<td>Base::ShaderInstanceBase</td>
</tr>
<tr>
<td>Base::ShaderServerBase</td>
</tr>
<tr>
<td>Base::ShaderVariableBase</td>
</tr>
<tr>
<td>Base::ShaderVariableInstanceBase</td>
</tr>
<tr>
<td>Base::ShaderVariationBase</td>
</tr>
<tr>
<td>Base::ShapeRendererBase</td>
</tr>
<tr>
<td>Base::StreamTextureSaverBase</td>
</tr>
<tr>
<td>Base::TextureBase</td>
</tr>
<tr>
<td>Base::TextureBase::MapInfo</td>
</tr>
<tr>
<td>Base::TransformDeviceBase</td>
</tr>
<tr>
<td>Base::VertexBufferBase</td>
</tr>
<tr>
<td>Base::VertexLayoutBase</td>
</tr>
<tr>
<td>BaseGameFeature::CategoryManager</td>
</tr>
<tr>
<td>BaseGameFeature::CategoryManager::Category</td>
</tr>
<tr>
<td>BaseGameFeature::CategoryManager::Entry</td>
</tr>
<tr>
<td>BaseGameFeature::EntityLoader</td>
</tr>
<tr>
<td>BaseGameFeature::EntityLoaderBase</td>
</tr>
<tr>
<td>BaseGameFeature::EntityManager</td>
</tr>
<tr>
<td>BaseGameFeature::EnvironmentLoader</td>
</tr>
<tr>
<td>Class</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>BaseGameFeature::EnvQueryManager</td>
</tr>
<tr>
<td>BaseGameFeature::FactoryManager</td>
</tr>
<tr>
<td>BaseGameFeature::FactoryManager::BluePrint</td>
</tr>
<tr>
<td>BaseGameFeature::FocusManager</td>
</tr>
<tr>
<td>BaseGameFeature::GlobalAttrsManager</td>
</tr>
<tr>
<td>BaseGameFeature::LevelLoader</td>
</tr>
<tr>
<td>BaseGameFeature::LoaderServer</td>
</tr>
<tr>
<td>BaseGameFeature::MoveFollow</td>
</tr>
<tr>
<td>BaseGameFeature::UserProfile</td>
</tr>
<tr>
<td>core</td>
</tr>
<tr>
<td>Core::CoreServer</td>
</tr>
<tr>
<td>Core::Factory</td>
</tr>
<tr>
<td>Core::Ptr</td>
</tr>
<tr>
<td>Core::RefCounted</td>
</tr>
<tr>
<td>Core::RefCountedList</td>
</tr>
<tr>
<td>Core::Rtti</td>
</tr>
<tr>
<td>Core::Singleton</td>
</tr>
<tr>
<td>Core::SysFunc</td>
</tr>
<tr>
<td>CoreGraphics::Adapter</td>
</tr>
<tr>
<td>CoreGraphics::AdapterInfo</td>
</tr>
<tr>
<td>CoreGraphics::AntiAliasQuality</td>
</tr>
<tr>
<td>CoreGraphics::BatchType</td>
</tr>
<tr>
<td>CoreGraphics::CPUIndexBuffer</td>
</tr>
<tr>
<td>CoreGraphics::CPUMemoryIndexBufferLoader</td>
</tr>
<tr>
<td>CoreGraphics::CPUMemoryVertexBufferLoader</td>
</tr>
<tr>
<td>CoreGraphics::CPUVertexBuffer</td>
</tr>
<tr>
<td>CoreGraphics::DisplayDevice</td>
</tr>
<tr>
<td>CoreGraphics::DisplayEvent</td>
</tr>
<tr>
<td>CoreGraphics::DisplayEventHandler</td>
</tr>
<tr>
<td>CoreGraphics::DisplayMode</td>
</tr>
<tr>
<td>CoreGraphics::ImageFileFormat</td>
</tr>
<tr>
<td>CoreGraphics::IndexBuffer</td>
</tr>
<tr>
<td>CoreGraphics::IndexType</td>
</tr>
<tr>
<td>CoreGraphics::MemoryIndexBufferLoader</td>
</tr>
<tr>
<td>CoreGraphics::MemoryVertexBufferLoader</td>
</tr>
<tr>
<td>CoreGraphics::Mesh</td>
</tr>
<tr>
<td>CoreGraphics::PixelFormat</td>
</tr>
<tr>
<td>CoreGraphics::PrimitiveGroup</td>
</tr>
<tr>
<td>CoreGraphics::PrimitiveTopology</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
<tr>
<td>CoreGraphics::RenderDevice</td>
</tr>
<tr>
<td>CoreGraphics::RenderEvent</td>
</tr>
<tr>
<td>CoreGraphics::RenderEventHandler</td>
</tr>
<tr>
<td>CoreGraphics::RenderTarget</td>
</tr>
<tr>
<td>CoreGraphics::Shader</td>
</tr>
<tr>
<td>CoreGraphics::ShaderFeature</td>
</tr>
<tr>
<td>CoreGraphics::ShaderInstance</td>
</tr>
<tr>
<td>CoreGraphics::ShaderServer</td>
</tr>
<tr>
<td>CoreGraphics::ShaderVariable</td>
</tr>
<tr>
<td>CoreGraphics::ShaderVariableInstance</td>
</tr>
<tr>
<td>CoreGraphics::ShaderVariation</td>
</tr>
<tr>
<td>CoreGraphics::ShapeRenderer</td>
</tr>
<tr>
<td>CoreGraphics::StreamAnimationLoader</td>
</tr>
<tr>
<td>CoreGraphics::StreamMeshLoader</td>
</tr>
<tr>
<td>CoreGraphics::StreamShaderLoader</td>
</tr>
<tr>
<td>CoreGraphics::StreamTextureLoader</td>
</tr>
<tr>
<td>CoreGraphics::StreamTextureSaver</td>
</tr>
<tr>
<td>CoreGraphics::Texture</td>
</tr>
<tr>
<td>CoreGraphics::ThreadSafeDisplayEventHandler</td>
</tr>
<tr>
<td>CoreGraphics::ThreadSafeRenderEventHandler</td>
</tr>
<tr>
<td>CoreGraphics::TransformDevice</td>
</tr>
<tr>
<td>CoreGraphics::VertexBuffer</td>
</tr>
<tr>
<td>CoreGraphics::VertexComponent</td>
</tr>
<tr>
<td>CoreGraphics::VertexLayout</td>
</tr>
<tr>
<td>CoreGraphics::VertexLayoutServer</td>
</tr>
<tr>
<td>Debug::CorePageHandler</td>
</tr>
<tr>
<td>Debug::DisplayPageHandler</td>
</tr>
<tr>
<td>Debug::IoPageHandler</td>
</tr>
<tr>
<td>Debug::MemoryPageHandler</td>
</tr>
<tr>
<td>Debug::MeshPageHandler</td>
</tr>
<tr>
<td>Debug::MiniDump</td>
</tr>
<tr>
<td>Debug::ScriptingPageHandler</td>
</tr>
<tr>
<td>Debug::ShaderPageHandler</td>
</tr>
<tr>
<td>Debug::TexturePageHandler</td>
</tr>
<tr>
<td>Direct3D9::D3D9DisplayDevice</td>
</tr>
<tr>
<td>Direct3D9::D3D9IndexBuffer</td>
</tr>
<tr>
<td>Direct3D9::D3D9MemoryIndexBufferLoader</td>
</tr>
<tr>
<td>Direct3D9::D3D9MemoryVertexBufferLoader</td>
</tr>
<tr>
<td>Direct3D9::D3D9RenderDevice</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>Direct3D9::D3D9RenderTarget</td>
</tr>
<tr>
<td>Direct3D9::D3D9Shader</td>
</tr>
<tr>
<td>Direct3D9::D3D9ShaderInstance</td>
</tr>
<tr>
<td>Direct3D9::D3D9ShaderServer</td>
</tr>
<tr>
<td>Direct3D9::D3D9ShaderVariable</td>
</tr>
<tr>
<td>Direct3D9::D3D9ShaderVariation</td>
</tr>
<tr>
<td>Direct3D9::D3D9ShapeRenderer</td>
</tr>
<tr>
<td>Direct3D9::D3D9StreamShaderLoader</td>
</tr>
<tr>
<td>Direct3D9::D3D9StreamTextureLoader</td>
</tr>
<tr>
<td>Direct3D9::D3D9StreamTextureSaver</td>
</tr>
<tr>
<td>Direct3D9::D3D9Texture</td>
</tr>
<tr>
<td>Direct3D9::D3D9Types</td>
</tr>
<tr>
<td>Direct3D9::D3D9VertexBuffer</td>
</tr>
<tr>
<td>Direct3D9::D3D9VertexLayout</td>
</tr>
<tr>
<td>Frame::FrameBatch</td>
</tr>
<tr>
<td>Frame::FramePass</td>
</tr>
<tr>
<td>Frame::FramePostEffect</td>
</tr>
<tr>
<td>Frame::FrameServer</td>
</tr>
<tr>
<td>Frame::FrameShader</td>
</tr>
<tr>
<td>Frame::FrameShaderLoader</td>
</tr>
<tr>
<td>Frame::LightingMode</td>
</tr>
<tr>
<td>Frame::SortingMode</td>
</tr>
<tr>
<td>Game::BaseGameFeatureUnit</td>
</tr>
<tr>
<td>Game::CoreFeature</td>
</tr>
<tr>
<td>Game::Entity</td>
</tr>
<tr>
<td>Game::FeatureUnit</td>
</tr>
<tr>
<td>Game::GameServer</td>
</tr>
<tr>
<td>Game::Manager</td>
</tr>
<tr>
<td>Game::PhysicsFeatureUnit</td>
</tr>
<tr>
<td>Game::Property</td>
</tr>
<tr>
<td>Graphics::ActorEntity</td>
</tr>
<tr>
<td>Graphics::CameraEntity</td>
</tr>
<tr>
<td>Graphics::Cell</td>
</tr>
<tr>
<td>Graphics::GraphicsEntity</td>
</tr>
<tr>
<td>Graphics::GraphicsServer</td>
</tr>
<tr>
<td>Graphics::ModelEntity</td>
</tr>
<tr>
<td>Graphics::QuadtreeStageBuilder</td>
</tr>
<tr>
<td>Graphics::SimpleStageBuilder</td>
</tr>
<tr>
<td>Class</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Graphics::Stage</td>
</tr>
<tr>
<td>Graphics::StageBuilder</td>
</tr>
<tr>
<td>Graphics::View</td>
</tr>
<tr>
<td>GraphicsFeature::CameraDistance</td>
</tr>
<tr>
<td>GraphicsFeature::CameraFocus</td>
</tr>
<tr>
<td>GraphicsFeature::CameraOrbit</td>
</tr>
<tr>
<td>GraphicsFeature::CameraReset</td>
</tr>
<tr>
<td>GraphicsFeature::GetGraphicsEntities</td>
</tr>
<tr>
<td>GraphicsFeature::GraphicsFeatureUnitUnit</td>
</tr>
<tr>
<td>GraphicsFeature::InputFocus</td>
</tr>
<tr>
<td>GraphicsFeature::SetVisible</td>
</tr>
<tr>
<td>Http::Base64Writer</td>
</tr>
<tr>
<td>Http::DefaultHttpRequestHandler</td>
</tr>
<tr>
<td>Http::HtmlElement</td>
</tr>
<tr>
<td>Http::HtmlPageWriter</td>
</tr>
<tr>
<td>Http::HttpMethod</td>
</tr>
<tr>
<td>Http::HttpRequest</td>
</tr>
<tr>
<td>Http::HttpRequestHandler</td>
</tr>
<tr>
<td>Http::HttpRequestReader</td>
</tr>
<tr>
<td>Http::HttpResponseWriter</td>
</tr>
<tr>
<td>Http::HttpServer</td>
</tr>
<tr>
<td>Http::HttpStatus</td>
</tr>
<tr>
<td>Input::GamePad</td>
</tr>
<tr>
<td>Input::InputEvent</td>
</tr>
<tr>
<td>Input::InputHandler</td>
</tr>
<tr>
<td>Input::InputPriority</td>
</tr>
<tr>
<td>Input::InputServer</td>
</tr>
<tr>
<td>Input::Key</td>
</tr>
<tr>
<td>Input::Keyboard</td>
</tr>
<tr>
<td>Input::Mouse</td>
</tr>
<tr>
<td>Input::MouseButton</td>
</tr>
<tr>
<td>Interface::CopyFile</td>
</tr>
<tr>
<td>Interface::CreateDirectory</td>
</tr>
<tr>
<td>Interface::DeleteDirectory</td>
</tr>
<tr>
<td>Interface::DeleteFile</td>
</tr>
<tr>
<td>Interface::IOMessage</td>
</tr>
<tr>
<td>Interface::MountZipArchive</td>
</tr>
<tr>
<td>Interface::ReadStream</td>
</tr>
<tr>
<td>Class</td>
</tr>
<tr>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Math::bbox</td>
</tr>
<tr>
<td>Math::ClipStatus</td>
</tr>
<tr>
<td>Math::float2</td>
</tr>
<tr>
<td>Math::float4</td>
</tr>
<tr>
<td>Math::line</td>
</tr>
<tr>
<td>Math::matrix44</td>
</tr>
<tr>
<td>Math::noise</td>
</tr>
<tr>
<td>Math::plane</td>
</tr>
<tr>
<td>Math::point</td>
</tr>
<tr>
<td>Math::polar</td>
</tr>
<tr>
<td>Math::quaternion</td>
</tr>
<tr>
<td>Math::rectangle&lt; TYPE &gt;</td>
</tr>
<tr>
<td>Math::sphere</td>
</tr>
<tr>
<td>Math::transform44</td>
</tr>
<tr>
<td>Math::vector</td>
</tr>
<tr>
<td>Memory::Heap</td>
</tr>
<tr>
<td>Memory::Memory</td>
</tr>
<tr>
<td>Memory::MemoryStatus</td>
</tr>
<tr>
<td>Messaging::AddAttachment</td>
</tr>
<tr>
<td>Messaging::AddSkin</td>
</tr>
<tr>
<td>Messaging::AnimationHotspotTriggered</td>
</tr>
<tr>
<td>Messaging::ApplyImpulseAtPos</td>
</tr>
<tr>
<td>Messaging::AsyncPort</td>
</tr>
<tr>
<td>Messaging::Dispatcher</td>
</tr>
<tr>
<td>Messaging::FadeAnimation</td>
</tr>
<tr>
<td>Messaging::GetActiveAnimation</td>
</tr>
<tr>
<td>Messaging::GetAnimationInfo</td>
</tr>
<tr>
<td>Messaging::GetAttachmentEntities</td>
</tr>
<tr>
<td>Messaging::GetHotspotTime</td>
</tr>
<tr>
<td>Messaging::GetJointMatrix</td>
</tr>
<tr>
<td>Messaging::GetPhysicsEntity</td>
</tr>
<tr>
<td>Messaging::Handler</td>
</tr>
<tr>
<td>Messaging::HasAttachment</td>
</tr>
<tr>
<td>Messaging::HideAttachment</td>
</tr>
<tr>
<td>Messaging::Id</td>
</tr>
<tr>
<td>Messaging::Message</td>
</tr>
<tr>
<td>Messaging::MessageReader</td>
</tr>
<tr>
<td>Messaging::MessageWriter</td>
</tr>
<tr>
<td>Messaging::MoveDirection</td>
</tr>
<tr>
<td>Messaging::MoveGoto</td>
</tr>
<tr>
<td>Messaging::MoveRotate</td>
</tr>
<tr>
<td>Messaging::MoveSetVelocity</td>
</tr>
<tr>
<td>Messaging::MoveStop</td>
</tr>
<tr>
<td>Messaging::MoveTurn</td>
</tr>
<tr>
<td>Messaging::Port</td>
</tr>
<tr>
<td>Messaging::RemAttachment</td>
</tr>
<tr>
<td>Messaging::RemSkin</td>
</tr>
<tr>
<td>Messaging::SetAnimation</td>
</tr>
<tr>
<td>Messaging::SetFadeAnimationMix</td>
</tr>
<tr>
<td>Messaging::SetTransform</td>
</tr>
<tr>
<td>Messaging::ShowAttachment</td>
</tr>
<tr>
<td>Messaging::UpdateAttachments</td>
</tr>
<tr>
<td>Messaging::UpdateTransform</td>
</tr>
<tr>
<td>Models::BinaryModelReader</td>
</tr>
<tr>
<td>Models::BinaryModelWriter</td>
</tr>
<tr>
<td>Models::CharacterNode</td>
</tr>
<tr>
<td>Models::CharacterNodeInstance</td>
</tr>
<tr>
<td>Models::ManagedModel</td>
</tr>
<tr>
<td>Models::Model</td>
</tr>
<tr>
<td>Models::ModelInstance</td>
</tr>
<tr>
<td>Models::ModelNode</td>
</tr>
<tr>
<td>Models::ModelNodeInstance</td>
</tr>
<tr>
<td>Models::ModelNodeType</td>
</tr>
<tr>
<td>Models::ModelReader</td>
</tr>
<tr>
<td>Models::ModelServer</td>
</tr>
<tr>
<td>Models::ModelWriter</td>
</tr>
<tr>
<td>Models::N2ModelReader</td>
</tr>
<tr>
<td>Models::ParticleSystemNode</td>
</tr>
<tr>
<td>Models::ParticleSystemNodeInstance</td>
</tr>
<tr>
<td>Models::ShapeNode</td>
</tr>
<tr>
<td>Models::ShapeNodeInstance</td>
</tr>
<tr>
<td>Models::SkinShapeNode</td>
</tr>
<tr>
<td>Models::SkinShapeNodeInstance</td>
</tr>
<tr>
<td>Models::StateNode</td>
</tr>
<tr>
<td>Models::StateNodeInstance</td>
</tr>
<tr>
<td>Models::StreamModelLoader</td>
</tr>
<tr>
<td>Models::TransformNode</td>
</tr>
<tr>
<td>Models::TransformNodeInstance</td>
</tr>
<tr>
<td>Class Name</td>
</tr>
<tr>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>Models::VisResolveContainer&lt; TYPE &gt;</td>
</tr>
<tr>
<td>Models::VisResolver</td>
</tr>
<tr>
<td>Models::XmlModelReader</td>
</tr>
<tr>
<td>Models::XmlModelWriter</td>
</tr>
<tr>
<td>Net::IpAddress</td>
</tr>
<tr>
<td>Net::Socket</td>
</tr>
<tr>
<td>Net::TcpClient</td>
</tr>
<tr>
<td>Net::TcpClientConnection</td>
</tr>
<tr>
<td>Net::TcpServer</td>
</tr>
<tr>
<td>PFeedbackLoop</td>
</tr>
<tr>
<td>PhysicsFeature::MouseGripperProperty</td>
</tr>
<tr>
<td>PhysicsFeature::PhysicsProperty</td>
</tr>
<tr>
<td>PIDFeedbackLoop</td>
</tr>
<tr>
<td>PQuatFeedbackLoop</td>
</tr>
<tr>
<td>PreShaders::BoxFilterKernel</td>
</tr>
<tr>
<td>PreShaders::GaussianBlur5x5FilterKernel</td>
</tr>
<tr>
<td>QuadTree</td>
</tr>
<tr>
<td>RenderUtil::MayaCameraUtil</td>
</tr>
<tr>
<td>Resources::DynamicMeshResourceLoader</td>
</tr>
<tr>
<td>Resources::ManagedMesh</td>
</tr>
<tr>
<td>Resources::ManagedResource</td>
</tr>
<tr>
<td>Resources::ManagedTexture</td>
</tr>
<tr>
<td>Resources::Resource</td>
</tr>
<tr>
<td>Resources::ResourceLoader</td>
</tr>
<tr>
<td>Resources::ResourceManager</td>
</tr>
<tr>
<td>Resources::ResourceMapper</td>
</tr>
<tr>
<td>Resources::ResourceSaver</td>
</tr>
<tr>
<td>Resources::SharedResourceServer</td>
</tr>
<tr>
<td>Resources::SimpleResourceMapper</td>
</tr>
<tr>
<td>Scripting::Arg</td>
</tr>
<tr>
<td>Scripting::ArgsBlock</td>
</tr>
<tr>
<td>Scripting::Command</td>
</tr>
<tr>
<td>Scripting::LuaServer</td>
</tr>
<tr>
<td>Scripting::ScriptServer</td>
</tr>
<tr>
<td>System::AppEntry</td>
</tr>
<tr>
<td>System::ByteOrder</td>
</tr>
<tr>
<td>System::Cpu</td>
</tr>
<tr>
<td>System::Win32Registry</td>
</tr>
<tr>
<td>Class</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Threading::Barrier</td>
</tr>
<tr>
<td>Threading::CriticalSection</td>
</tr>
<tr>
<td>Threading::Event</td>
</tr>
<tr>
<td>Threading::Interlocked</td>
</tr>
<tr>
<td>Threading::SafePriorityQueue&lt; PRITYPE, TYPE &gt;</td>
</tr>
<tr>
<td>Threading::SafeQueue&lt; TYPE &gt;</td>
</tr>
<tr>
<td>Threading::Thread</td>
</tr>
<tr>
<td>Threading::ThreadLocalPtr&lt; T &gt;</td>
</tr>
<tr>
<td>Timing::CalendarTime</td>
</tr>
<tr>
<td>Timing::InputTimeSource</td>
</tr>
<tr>
<td>Timing::SystemTimeSource</td>
</tr>
<tr>
<td>Timing::TimeManager</td>
</tr>
<tr>
<td>Timing::Timer</td>
</tr>
<tr>
<td>Timing::TimeSource</td>
</tr>
<tr>
<td>Timing::TimingTimeSource</td>
</tr>
<tr>
<td>Util::Array&lt; TYPE &gt;</td>
</tr>
<tr>
<td>Util::Atom&lt; TYPE &gt;</td>
</tr>
<tr>
<td>Util::Blob</td>
</tr>
<tr>
<td>Util::CharEnhancementUtil</td>
</tr>
<tr>
<td>Util::CmdLineArgs</td>
</tr>
<tr>
<td>Util::Crc</td>
</tr>
<tr>
<td>Util::Dictionary&lt; KEYTYPE, VALUETYPE &gt;</td>
</tr>
<tr>
<td>Util::FixedArray&lt; TYPE &gt;</td>
</tr>
<tr>
<td>Util::FixedTable&lt; TYPE &gt;</td>
</tr>
<tr>
<td>Util::FourCC</td>
</tr>
<tr>
<td>Util::Guid</td>
</tr>
<tr>
<td>Util::HashTable&lt; KEYTYPE, VALUETYPE &gt;</td>
</tr>
<tr>
<td>Util::KeyValuePair&lt; KEYTYPE, VALUETYPE &gt;</td>
</tr>
<tr>
<td>Util::LightFlickerUtil</td>
</tr>
<tr>
<td>Util::List&lt; TYPE &gt;</td>
</tr>
<tr>
<td>Util::List&lt; TYPE &gt;::Iterator</td>
</tr>
<tr>
<td>Util::Proxy&lt; TYPE &gt;</td>
</tr>
<tr>
<td>Util::QuadTree&lt; TYPE &gt;::Node</td>
</tr>
<tr>
<td>Util::Queue&lt; TYPE &gt;</td>
</tr>
<tr>
<td>Util::SegmentedGfxUtil</td>
</tr>
<tr>
<td>Util::SimpleTree&lt; VALUETYPE &gt;</td>
</tr>
<tr>
<td>Util::SimpleTree&lt; VALUETYPE &gt;::Node</td>
</tr>
<tr>
<td>Util::Stack&lt; TYPE &gt;</td>
</tr>
<tr>
<td>Class</td>
</tr>
<tr>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Util::String</td>
</tr>
<tr>
<td>Util::StringAtom</td>
</tr>
<tr>
<td>Util::Variant</td>
</tr>
<tr>
<td>Win32::SysFunc</td>
</tr>
<tr>
<td>Win32::Win32Barrier</td>
</tr>
<tr>
<td>Win32::Win32CalendarTime</td>
</tr>
<tr>
<td>Win32::Win32ConsoleHandler</td>
</tr>
<tr>
<td>Win32::Win32CriticalSection</td>
</tr>
<tr>
<td>Win32::Win32DisplayDevice</td>
</tr>
<tr>
<td>Win32::Win32Event</td>
</tr>
<tr>
<td>Win32::Win32FileTime</td>
</tr>
<tr>
<td>Win32::Win32FSWrapper</td>
</tr>
<tr>
<td>Win32::Win32Guid</td>
</tr>
<tr>
<td>Win32::Win32Heap</td>
</tr>
<tr>
<td>Win32::Win32InputDisplayEventHandler</td>
</tr>
<tr>
<td>Win32::Win32InputServer</td>
</tr>
<tr>
<td>Win32::Win32Interlocked</td>
</tr>
<tr>
<td>Win32::Win32IpAddress</td>
</tr>
<tr>
<td>Win32::Win32MiniDump</td>
</tr>
<tr>
<td>Win32::Win32Mouse</td>
</tr>
<tr>
<td>Win32::Win32Socket</td>
</tr>
<tr>
<td>Win32::Win32Thread</td>
</tr>
<tr>
<td>Win32::Win32Timer</td>
</tr>
<tr>
<td>XInput::XInputGamePad</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

#include <angularpfeedbackloop.h>
Detailed Description

A proportional feedback loop with correct angular interpolation.

(C) 2004 RadonLabs 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>virtual <code>~Application ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>void <code>SetCompanyName (const Util::String &amp;n)</code></td>
<td>set company name</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetCompanyName ()</code> const</td>
<td>get company name</td>
</tr>
<tr>
<td>void <code>SetAppName (const Util::String &amp;n)</code></td>
<td>set application name</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetAppName ()</code> const</td>
<td>get application name</td>
</tr>
<tr>
<td>void <code>SetCmdLineArgs (const Util::CmdLineArgs &amp;a)</code></td>
<td>set command line args</td>
</tr>
<tr>
<td>const Util::CmdLineArgs &amp; <code>GetCmdLineArgs ()</code> const</td>
<td>get command line args</td>
</tr>
<tr>
<td>virtual bool <code>Open ()</code></td>
<td>open the application</td>
</tr>
<tr>
<td>virtual void <code>Close ()</code></td>
<td>close the application</td>
</tr>
<tr>
<td>virtual void <code>Exit ()</code></td>
<td>exit the application, call right before leaving main()</td>
</tr>
<tr>
<td>virtual void <code>Run ()</code></td>
<td>run the application, return when user wants to exit</td>
</tr>
<tr>
<td>bool <code>IsOpen ()</code> const</td>
<td>return true if app is open</td>
</tr>
<tr>
<td>int <code>GetReturnCode ()</code> const</td>
<td>get the return code</td>
</tr>
</tbody>
</table>
Protected Member Functions

void SetReturnCode (int c)

set return code
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

App::AsyncRenderApplication
App::AsyncRenderApplication Class Reference

#include <asyncrenderapplication.h>

Inheritance diagram for App::AsyncRenderApplication:
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><strong>AsyncRenderApplication ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~AsyncRenderApplication ()</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>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 SetAppName (const Util::String &amp;n)</td>
<td>set application name</td>
</tr>
<tr>
<td>const Util::String &amp; GetAppName () const</td>
<td>get application name</td>
</tr>
<tr>
<td>void SetCmdLineArgs (const Util::CmdLineArgs &amp;a)</td>
<td>set command line args</td>
</tr>
<tr>
<td>const Util::CmdLineArgs &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>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>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>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>
App::AsyncViewerApplication
App::AsyncViewerApplication Class Reference

#include <asyncviewerapplication.h>

Inheritance diagram for App::AsyncViewerApplication:
Detailed Description

Derived from AsyncRenderApplication, 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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsyncViewerApplication()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~AsyncViewerApplication()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual bool Open()</code></td>
<td>Open the application</td>
</tr>
<tr>
<td><code>virtual void Close()</code></td>
<td>Close the application</td>
</tr>
<tr>
<td><code>virtual void Run()</code></td>
<td>Run the application</td>
</tr>
<tr>
<td><code>void SetCompanyName(const Util::String &amp;n)</code></td>
<td>Set company name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetCompanyName()</code></td>
<td>Get company name</td>
</tr>
<tr>
<td><code>void SetAppName(const Util::String &amp;n)</code></td>
<td>Set application name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetAppName()</code></td>
<td>Get application name</td>
</tr>
<tr>
<td><code>void SetCmdLineArgs(const Util::CmdLineArgs &amp;a)</code></td>
<td>Set command line args</td>
</tr>
<tr>
<td><code>const Util::CmdLineArgs &amp; GetCmdLineArgs()</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()</code></td>
<td>Return true if app is open</td>
</tr>
<tr>
<td><code>int GetReturnCode()</code></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 <strong>OnProcessInput</strong> ()</td>
<td>process input (called before rendering)</td>
</tr>
<tr>
<td>virtual void <strong>OnUpdateFrame</strong> ()</td>
<td>update world</td>
</tr>
<tr>
<td>void <strong>SetQuitRequested</strong> (bool b)</td>
<td>set quit requested flag</td>
</tr>
<tr>
<td>bool <strong>IsQuitRequested</strong> () const</td>
<td>return true if quit requested</td>
</tr>
<tr>
<td>virtual void <strong>OnConfigureDisplay</strong> ()</td>
<td>called to configure display</td>
</tr>
<tr>
<td><strong>Timing::Time</strong> <strong>GetTime</strong> () const</td>
<td>get current absolute time</td>
</tr>
<tr>
<td><strong>Timing::Time</strong> <strong>GetFrameTime</strong> () const</td>
<td>get current frame time</td>
</tr>
<tr>
<td>void <strong>SetReturnCode</strong> (int c)</td>
<td>set return code</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Tue Feb 19 12:16:41 2008
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

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>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ConsoleApplication()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~ConsoleApplication()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual bool Open()</code></td>
<td>Open the application</td>
</tr>
<tr>
<td><code>virtual void Close()</code></td>
<td>Close the application</td>
</tr>
<tr>
<td><code>void SetCompanyName(const Util::String &amp;n)</code></td>
<td>Set company name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetCompanyName() const</code></td>
<td>Get company name</td>
</tr>
<tr>
<td><code>void SetAppName(const Util::String &amp;n)</code></td>
<td>Set application name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetAppName() const</code></td>
<td>Get application name</td>
</tr>
<tr>
<td><code>void SetCmdLineArgs(const Util::CmdLineArgs &amp;a)</code></td>
<td>Set command line args</td>
</tr>
<tr>
<td><code>const Util::CmdLineArgs &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>virtual void Run()</code></td>
<td>Run the application, return when user wants to exit</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

void SetReturnCode (int c)

set return code
- 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:
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 gamestate (such as level gamestates or only gui gamestates).

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GameApplication()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~GameApplication()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>virtual bool Open()</code></td>
<td>open the application</td>
</tr>
<tr>
<td><code>virtual void Close()</code></td>
<td>close the application</td>
</tr>
<tr>
<td><code>virtual void Run()</code></td>
<td>run the application</td>
</tr>
<tr>
<td><code>void AddStateHandler(const Ptr&lt;StateHandler&gt; &amp;state)</code></td>
<td>add an application state handler</td>
</tr>
<tr>
<td><code>const Ptr&lt;StateHandler&gt; &amp; FindStateHandlerByName(const Util::String &amp;stateName)</code></td>
<td>find a state handler by name</td>
</tr>
<tr>
<td><code>const Ptr&lt;StateHandler&gt; &amp; GetCurrentStateHandler()</code></td>
<td>return pointer to current state handler</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetCurrentState()</code></td>
<td>return state handler of current state</td>
</tr>
<tr>
<td><code>int GetNumStates()</code></td>
<td>get number of application states</td>
</tr>
<tr>
<td><code>const Ptr&lt;StateHandler&gt; &amp; GetStateHandlerAt(int index)</code></td>
<td>get state handler at index</td>
</tr>
<tr>
<td><code>void RequestState(const Util::String &amp;stateName)</code></td>
<td>request a new state which will be applied at the end of the frame</td>
</tr>
<tr>
<td><code>void SetCompanyName(const Util::String &amp;n)</code></td>
<td>set company name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetCompanyName()</code></td>
<td>get company name</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void SetAppName (const Util::String &amp;n)</code></td>
<td>set application name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetAppName () const</code></td>
<td>get application name</td>
</tr>
<tr>
<td><code>void SetCmdLineArgs (const Util::CmdLineArgs &amp;a)</code></td>
<td>set command line args</td>
</tr>
<tr>
<td><code>const Util::CmdLineArgs &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>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void</td>
<td><code>SetupStateHandlers()</code></td>
<td>setup application state handlers</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>CleanupStateHandlers()</code></td>
<td>cleanup application state handlers</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>SetupGameFeatures()</code></td>
<td>setup game features</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>CleanupGameFeatures()</code></td>
<td>cleanup game features</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>DoStateTransition()</code></td>
<td>perform a state transition</td>
</tr>
<tr>
<td>void</td>
<td><code>SetState(const String &amp;s)</code></td>
<td>set an application state</td>
</tr>
<tr>
<td>void</td>
<td><code>SetReturnCode(int c)</code></td>
<td>set return code</td>
</tr>
</tbody>
</table>
### Protected Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>requestedState</td>
</tr>
<tr>
<td>gameServer</td>
<td>state handlers</td>
</tr>
<tr>
<td>CoreFeatureUnit</td>
<td>coreFeature</td>
</tr>
<tr>
<td>game server</td>
<td>default game features</td>
</tr>
</tbody>
</table>

- *requestedState*: State handlers
- *gameServer*: Game server
- *coreFeature*: Default game features
Member Function Documentation

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 Ptr<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(const 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:

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

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

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

```cpp
const 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.

```cpp
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.

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

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.

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

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.

```cpp
void
App::GameApplication::SetupGameFeatures() [protected, virtual]
```
setup game features

Setup new game features which should be used by this application. Overwrite if other features or more features 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 String & s) [protected]
```

set an application state

Set a new application state. This method will call `DoStateTransition()`.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

App::GameStateHandler
App::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

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><strong>SetupMode</strong></td>
</tr>
<tr>
<td></td>
<td><em>setup modes</em></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GameStateHandler</strong></td>
<td><strong>()</strong> constructor</td>
</tr>
<tr>
<td><strong>~GameStateHandler</strong></td>
<td><strong>()</strong> destructor</td>
</tr>
<tr>
<td>virtual void <strong>OnStateEnter</strong></td>
<td>(const Util::String &amp;prevState) <strong>called when the state represented by this state handler is entered</strong></td>
</tr>
<tr>
<td>virtual void <strong>OnStateLeave</strong></td>
<td>(const Util::String &amp;nextState) <strong>called when the state represented by this state handler is left</strong></td>
</tr>
<tr>
<td>virtual <strong>Util::String</strong></td>
<td><strong>OnFrame</strong> <strong>()</strong> <strong>called each frame as long as state is current, return new state</strong></td>
</tr>
<tr>
<td>void <strong>SetSetupMode</strong></td>
<td>(<strong>SetupMode</strong> mode) <strong>set the setup mode</strong></td>
</tr>
<tr>
<td><strong>SetupMode</strong></td>
<td><strong>GetSetupMode</strong> <strong>()</strong> <strong>const</strong> <strong>get the setup mode</strong></td>
</tr>
<tr>
<td>void <strong>SetLevelName</strong></td>
<td>(<strong>const Util::String &amp;n</strong>) <strong>set level filename, required by setup mode LoadLevel</strong></td>
</tr>
<tr>
<td><strong>const Util::String &amp;</strong></td>
<td><strong>GetLevelName</strong> <strong>()</strong> <strong>const</strong> <strong>get level name</strong></td>
</tr>
<tr>
<td>void <strong>SetSaveGame</strong></td>
<td>(<strong>const Util::String &amp;n</strong>) <strong>set save game name, required by setup mode LoadSaveGame</strong></td>
</tr>
<tr>
<td><strong>const Util::String &amp;</strong></td>
<td><strong>GetSaveGame</strong> <strong>()</strong> <strong>const</strong> <strong>get save game name</strong></td>
</tr>
</tbody>
</table>
Member Function Documentation

void App::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

void App::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

Util::String App::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
The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:41 2008
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:

```
App::Application

App::RenderApplication

App::ViewerApplication
```
Detailed Description

**Base** class for Nebula3 applications with 3d rendering capability.

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

<table>
<thead>
<tr>
<th>Function Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RenderApplication ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>virtual ~RenderApplication ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>virtual bool Open ()</code></td>
<td>open the application</td>
</tr>
<tr>
<td><code>virtual void Close ()</code></td>
<td>close the application</td>
</tr>
<tr>
<td><code>virtual void Run ()</code></td>
<td>run the application</td>
</tr>
<tr>
<td><code>void SetCompanyName (const Util::String &amp;n)</code></td>
<td>set company name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetCompanyName () const</code></td>
<td>get company name</td>
</tr>
<tr>
<td><code>void SetAppName (const Util::String &amp;n)</code></td>
<td>set application name</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetAppName () const</code></td>
<td>get application name</td>
</tr>
<tr>
<td><code>void SetCmdLineArgs (const Util::CmdLineArgs &amp;a)</code></td>
<td>set command line args</td>
</tr>
<tr>
<td><code>const Util::CmdLineArgs &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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetQuitRequested(bool b)</code></td>
<td>set quit requested flag</td>
</tr>
<tr>
<td><code>bool IsQuitRequested()</code> const</td>
<td>return true if quit requested</td>
</tr>
<tr>
<td><code>virtual void OnConfigureDisplayDevice()</code></td>
<td>called to configure display device</td>
</tr>
<tr>
<td><code>virtual void OnProcessInput()</code></td>
<td>process input (called before rendering)</td>
</tr>
<tr>
<td><code>virtual void OnUpdateFrame()</code></td>
<td>update world</td>
</tr>
<tr>
<td><code>virtual void OnRenderFrame()</code></td>
<td>render current frame</td>
</tr>
<tr>
<td><code>Timing::Time GetTime()</code> const</td>
<td>get current absolute time</td>
</tr>
<tr>
<td><code>Timing::Time GetFrameTime()</code> const</td>
<td>get current frame time</td>
</tr>
<tr>
<td><code>void SetReturnCode(int c)</code></td>
<td>set return code</td>
</tr>
</tbody>
</table>
App::ViewerApplication
#include <viewerapplication.h>

Inheritance diagram for App::ViewerApplication:
Detailed Description

Nebula3's default viewer application.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</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>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 SetAppName (const Util::String &amp;n)</td>
<td>set application name</td>
</tr>
<tr>
<td>const Util::String &amp; GetAppName () const</td>
<td>get application name</td>
</tr>
<tr>
<td>void SetCmdLineArgs (const Util::CmdLineArgs &amp;a)</td>
<td>set command line args</td>
</tr>
<tr>
<td>const Util::CmdLineArgs &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 Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| virtual void  | **OnProcessInput** ()  
*process input (called before rendering)* |
| virtual void  | **OnRenderFrame** ()  
*render current frame* |
| void          | **SetQuitRequested** (bool b)  
*set quit requested flag* |
| bool          | **IsQuitRequested** ()  
*return true if quit requested* |
| virtual void  | **OnConfigureDisplayDevice** ()  
*called to configure display device* |
| virtual void  | **OnUpdateFrame** ()  
*update world* |
| Timing::Time  | **GetTime** ()  
*get current absolute time* |
| Timing::Time  | **GetFrameTime** ()  
*get current frame time* |
| void          | **SetReturnCode** (int c)  
*set return code* |
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
AsyncGraphics::AsyncGraphicsHandler
AsyncGraphics::AsyncGraphicsHandler
Class Reference

#include <asyncgraphicshandler.h>

Inheritance diagram for AsyncGraphics::AsyncGraphicsHandler:
**Detailed Description**

Runs in the graphics thread context, sets up the graphics runtime environment and processes messages to the graphics thread.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsyncGraphicsHandler()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~AsyncGraphicsHandler()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <code>Open()</code></td>
<td>Open the handler</td>
</tr>
<tr>
<td>virtual void <code>Close()</code></td>
<td>Close the handler</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>Do per-frame work</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>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</td>
<td><code>IsA</code> (const <code>Util::String</code> &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><code>IsA</code> (const <code>Util::FourCC</code> &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 <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 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 AsyncGraphics::AsyncGraphicsHandler::DoWork() [virtual]

 do per-frame work

This is the per-frame method which implements the asynchronous render-loop.

Reimplemented from `Messaging::Handler`.
```
```cpp
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.
```
```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.
AsyncGraphics::AsyncGraphicsInterface
AsyncGraphics::AsyncGraphicsInterface
Class Reference

#include <asyncgraphicsinterface.h>

Inheritance diagram for AsyncGraphics::AsyncGraphicsInterface:
Detailed Description

Implements the asynchronous interface to the Graphics subsystem. Usually the application doesn't call methods on the AsyncGraphicsInterface directly, but instead uses one of the AsyncGraphics proxy 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><code>AsyncGraphicsInterface ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~AsyncGraphicsInterface ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Open ()</code></td>
<td>Open the interface object</td>
</tr>
<tr>
<td><code>SetName (const Util::String &amp;n)</code></td>
<td>Set the name of the async port (required)</td>
</tr>
<tr>
<td><code>GetName () const</code></td>
<td>Get the name of the async port</td>
</tr>
<tr>
<td><code>SetThreadPriority (Threading::Thread::Priority pri)</code></td>
<td>Set optional thread priority</td>
</tr>
<tr>
<td><code>GetThreadPriority () const</code></td>
<td>Get optional thread priority</td>
</tr>
<tr>
<td><code>SetThreadStackSize (unsigned int s)</code></td>
<td>Set optional thread stack size</td>
</tr>
<tr>
<td><code>GetThreadStackSize () const</code></td>
<td>Get optional thread stack size</td>
</tr>
<tr>
<td><code>SetWaitForMessages (bool b)</code></td>
<td>Wait-for-messages or run-continuously? (default is wait-for-message)</td>
</tr>
<tr>
<td><code>GetWaitForMessages () const</code></td>
<td>Get wait-for-message mode</td>
</tr>
<tr>
<td><code>AttachHandler (const Ptr&lt; Handler &gt; &amp;h)</code></td>
<td>Attach a handler to the port (call before open!)</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; Message &gt; &amp;msg)</code></td>
<td>Send an asynchronous message to the port</td>
</tr>
<tr>
<td><code>SendWait (const Ptr&lt; Message &gt; &amp;msg)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Wait (const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>wait for a message to be handled</td>
</tr>
<tr>
<td><code>Peek (const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>peek a message whether it has been handled</td>
</tr>
<tr>
<td><code>Cancel (const Ptr&lt;Message&gt; &amp;msg)</code></td>
<td>cancel a pending message</td>
</tr>
<tr>
<td><code>Flush ()</code></td>
<td>wait until all pending messages have been handled</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</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
virtual void OnCreateHandlers ()
```

derive in subclass, create and attach handlers from here
Member Function Documentation

void Messaging::AsyncPort::AttachHandler(const Ptr<Handler> &h) [inherited]

attach a handler to the port (call before open!)

Called by **OnCreateHandlers()** method of subclass to attach a handler to the port.

void Messaging::AsyncPort::Close()

close the async port

Closes the async port.

void Messaging::AsyncPort::Send(const Ptr<Message> &msg) [inherited]

send an asynchronous message to the port

Handle an asynchronous message and return immediately. If the caller expects any results from the message he can poll with the **AsyncPort:: Peek()** method, or he may wait for the message to be handled with the **AsyncPort:: Wait()** method.

void Messaging::AsyncPort::SendWait(const Ptr<Message> &msg) [inherited]

send a message and wait for completion

Send an asynchronous message and wait until the message has been handled.
void Messaging::AsyncPort::Wait(const Ptr<Message> &msg) [inherited]

wait for a message to be handled

This method will wait until a message has been handled. If the caller expects any return arguments from the message handling it can use this method to wait for the results.

bool Messaging::AsyncPort::Peek(const Ptr<Message> &msg) [inherited]

peek a message whether it has been handled

This method peeks whether a message has already been handled. If the caller expects any return arguments from the message handling it can use this message to check whether the results are ready using this non-blocking method. The caller can also wait for the results to become ready using the `Wait()` method.

void Messaging::AsyncPort::Cancel(const Ptr<Message> &msg) [inherited]

cancel a pending message

This method will cancel a pending message.

void Messaging::AsyncPort::Flush() [inherited]

wait until all pending messages have been handled

This method will wait until ALL pending messages have been handled. Note that this method will be called automatically before the AsyncPort shutdown.

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

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.
AsyncGraphics::CameraEntityProxy
AsyncGraphics::CameraEntityProxy
Class Reference

#include <cameraentityproxy.h>

Inheritance diagram for AsyncGraphics::CameraEntityProxy:
Detailed Description

Client-side proxy of a Graphics::CameraEntity. 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 then 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><strong>CameraEntityProxy ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~CameraEntityProxy ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>SetViewName</strong> (const Util::StringAtom &amp;n)</td>
<td>set the view name which should use this camera</td>
</tr>
<tr>
<td>const Util::StringAtom &amp; <strong>GetViewName</strong> () const</td>
<td>get the view name which should use this camera</td>
</tr>
<tr>
<td>void <strong>SetupPerspectiveFov</strong> (float fov, float aspect, float zNear, float zFar)</td>
<td>setup camera for perspective field-of-view projection transform</td>
</tr>
<tr>
<td>void <strong>SetupOrthogonal</strong> (float w, float h, float zNear, float zFar)</td>
<td>setup camera for orthogonal projection transform</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; <strong>GetProjTransform</strong> () const</td>
<td>get projection matrix</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; <strong>GetViewTransform</strong> () const</td>
<td>get view transform (inverse transform)</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; <strong>GetViewProjTransform</strong> ()</td>
<td>get view projection matrix (non-const!)</td>
</tr>
<tr>
<td>bool <strong>IsPerspective</strong> () const</td>
<td>return true if this is a perspective projection</td>
</tr>
<tr>
<td>bool <strong>IsOrthogonal</strong> () const</td>
<td>return true if this is an orthogonal transform</td>
</tr>
<tr>
<td>float <strong>GetZNear</strong> () const</td>
<td>get near plane distance</td>
</tr>
<tr>
<td>float <strong>GetZFar</strong> () const</td>
<td>get far plane distance</td>
</tr>
<tr>
<td>float <strong>GetFov</strong> () const</td>
<td>get field-of-view (only if perspective)</td>
</tr>
<tr>
<td>float <strong>GetAspect</strong> () const</td>
<td>get aspectration (only if perspective)</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetNearWidth() const</code></td>
<td>Get width of near plane</td>
</tr>
<tr>
<td><code>GetNearHeight() const</code></td>
<td>Get height of near plane</td>
</tr>
<tr>
<td><code>GetFarWidth() const</code></td>
<td>Get width of far plane</td>
</tr>
<tr>
<td><code>GetFarHeight() const</code></td>
<td>Get height of far plane</td>
</tr>
<tr>
<td><code>IsValid() const</code></td>
<td>Return true if entity is valid (is attached to a stage)</td>
</tr>
<tr>
<td><code>GetType() const</code></td>
<td>Get the entity type</td>
</tr>
<tr>
<td><code>SetTransform(const Math::matrix44 &amp;m)</code></td>
<td>Set the entity's world space transform</td>
</tr>
<tr>
<td><code>GetTransform() const</code></td>
<td>Get the entity's world space transform</td>
</tr>
<tr>
<td><code>SetVisible(bool b)</code></td>
<td>Set the entity's visibility</td>
</tr>
<tr>
<td><code>IsVisible() const</code></td>
<td>Return true if entity is set to visible</td>
</tr>
<tr>
<td><code>GetStageProxy() const</code></td>
<td>Get the stage proxy this entity is attached to</td>
</tr>
<tr>
<td><code>IsAttachedToStageProxy() const</code></td>
<td>Return true if entity is attached to stage</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</td>
</tr>
<tr>
<td></td>
<td>Function</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td></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></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></td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td></td>
<td><strong>GetClassFourCC</strong> () 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>virtual void</th>
<th><strong>Setup</strong> (const Ptr&lt; StageProxy &gt; &amp;stage)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>called by stage when entity should setup itself</em></td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnTransformChanged</strong> ()</td>
</tr>
<tr>
<td></td>
<td><em>called when transform matrix changed</em></td>
</tr>
<tr>
<td>void</td>
<td><strong>UpdateViewProjMatrix</strong> ()</td>
</tr>
<tr>
<td></td>
<td><em>update the view projection matrix</em></td>
</tr>
<tr>
<td>void</td>
<td><strong>SetType</strong> (Graphics::GraphicsEntity::Type t)</td>
</tr>
<tr>
<td></td>
<td><em>set graphics entity type, called from constructor of subclass</em></td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>Discard</strong> ()</td>
</tr>
<tr>
<td></td>
<td><em>called by stage when entity should discard itself</em></td>
</tr>
<tr>
<td>void</td>
<td><strong>ValidateEntityHandle</strong> (bool wait=false)</td>
</tr>
<tr>
<td></td>
<td><em>validate entity handle</em></td>
</tr>
<tr>
<td>void</td>
<td><strong>SendCreateMsg</strong> (const Ptr&lt; CreateGraphicsEntity &gt; &amp;msg)</td>
</tr>
<tr>
<td></td>
<td><em>send off a specific create message from the subclass</em></td>
</tr>
<tr>
<td>void</td>
<td><strong>SendMsg</strong> (const Ptr&lt; GraphicsEntityMsg &gt; &amp;msg)</td>
</tr>
<tr>
<td></td>
<td><em>send a message to the server-side graphics entity</em></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsEntityHandleValid</strong> () const</td>
</tr>
<tr>
<td></td>
<td><em>test if the entity handle is valid</em></td>
</tr>
</tbody>
</table>
void AsyncGraphics::CameraEntityProxy::SetupPerspectiveFov(float fov_,
float aspect_,
float zNear_,
float zFar_)

setup camera for perspective field-of-view projection transform

Setup camera as perspective projection. NOTE: this method is a copy of CameraEntity::SetupPerspectiveFov()!

void AsyncGraphics::CameraEntityProxy::SetupOrthogonal(float w,
float h,
float zNear_,
float zFar_)

setup camera for orthogonal projection transform

Setup camera as orthogonal projection. NOTE: this method is a copy of CameraEntity::SetupOrthogonal()!

const Math::matrix44 &
AsyncGraphics::CameraEntityProxy::GetViewTransform() const [inline]

get view transform (inverse transform)

NOTE: The matrix returned here may be off a little bit from view matrix used in the render thread because the game thread may run at a slower frame rate then rendering! It is not possible to get an exact view matrix outside of the render thread.

const Math::matrix44 &
AsyncGraphics::CameraEntityProxy::GetViewProjTransform() [inline]

get view projection matrix (non-const!)
NOTE: The matrix returned here may be off a little bit from viewProj matrix used in the render thread because the game thread may run at a slower frame rate then rendering! It is not possible to get an exact viewProj matrix outside of the render thread.

```cpp
void AsyncGraphics::CameraEntityProxy::Setup(
  const Ptr<StageProxy>& stageProxy_)
  [protected, virtual]

called by stage when entity should setup itself

Setup the server-side camera entity.

Reimplemented from AsyncGraphics::GraphicsEntityProxy.

void AsyncGraphics::CameraEntityProxy::OnTransformChanged()
  [protected, virtual]

called when transform matrix changed

We need to keep track of modifications of the transformation matrix.

Reimplemented from AsyncGraphics::GraphicsEntityProxy.

void AsyncGraphics::CameraEntityProxy::UpdateViewProjMatrix()
  [protected]

update the view projection matrix

Updates the view-projection matrix.

void AsyncGraphics::GraphicsEntityProxy::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(). If the server-side entity hasn't been created yet, this method needs to wait for completion, since the entity handle won't be available yet!

```
validate entity handle

Tries to validate the entity handle by checking whether the pending creation message has already been handled by the server side. If the wait flag is true, the method will wait until the entity has been created on the server side. The default is to NOT wait. If the graphics entity handle becomes valid, all pending messages will be sent off to the

```cpp
void AsyncGraphics::GraphicsEntityProxy::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 AsyncGraphics::GraphicsEntityProxy::SendMsg (const Ptr<GraphicsEntityMsg> &msg ) [protected, inherited]
```

send a message to the server-side graphics entity

Send a generic GraphicsEntityMsg to the server-side entity. Do not initialize the entity handle of the message before calling this method, since the server-side entity doesn't have to exist yet. In this case, the message will be queued up and sent off as soon as the server-side entity becomes valid.

```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.

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.
AsyncGraphics::DisplayProxy
# AsyncGraphics::DisplayProxy Class Reference

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

Inheritance diagram for AsyncGraphics::DisplayProxy:

![Inheritance Diagram](image.png)
Detailed Description

The **DisplayProxy** object is used to access DisplayDevice functionality from a different thread. Usually only the main thread creates a display proxy object.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Public Member Functions</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DisplayProxy ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~DisplayProxy ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void SetAdapter (CoreGraphics::Adapter::Code set display adapter to use)</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::Adapter::Code GetAdapter () const</td>
<td>get display adapter</td>
</tr>
<tr>
<td>void SetDisplayMode (const CoreGraphics::DisplayMode &amp;m)</td>
<td>set display mode (make sure the display mode is supported!)</td>
</tr>
<tr>
<td>const CoreGraphics::DisplayMode &amp; GetDisplayMode () const</td>
<td>get display mode</td>
</tr>
<tr>
<td>void SetAntiAliasQuality (CoreGraphics::AntiAliasQuality::Code set antialias quality)</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::AntiAliasQuality::Code GetAntiAliasQuality () const</td>
<td>get antialias quality</td>
</tr>
<tr>
<td>void SetFullscreen (bool b)</td>
<td>set windowed/fullscreen mode</td>
</tr>
<tr>
<td>bool IsFullscreen () const</td>
<td>get windowed/fullscreen mode</td>
</tr>
<tr>
<td>void SetDisplayModeSwitchEnabled (bool b)</td>
<td>enable display mode switch when running fullscreen (true);</td>
</tr>
<tr>
<td>bool IsDisplayModeSwitchEnabled () const</td>
<td>is display mode switch enabled for fullscreen?</td>
</tr>
<tr>
<td>void SetTripleBufferingEnabled (bool b)</td>
<td>enable triple buffer for fullscreen (default is double buffer);</td>
</tr>
<tr>
<td>bool IsTripleBufferingEnabled () const</td>
<td>is triple buffer enabled for fullscreen?</td>
</tr>
<tr>
<td>void SetAlwaysOnTop (bool b)</td>
<td>set always-on-top behaviour</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool IsAlwaysOnTop (const)</code></td>
<td>Check if always-on-top behaviour is enabled</td>
</tr>
<tr>
<td><code>void SetVerticalSyncEnabled (bool b)</code></td>
<td>Turn vertical sync on or off</td>
</tr>
<tr>
<td><code>bool IsVerticalSyncEnabled (const)</code></td>
<td>Check if vertical sync flag is enabled</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 Open ()</code></td>
<td>Open the display (waits for completion)</td>
</tr>
<tr>
<td><code>void Close ()</code></td>
<td>Close the display (waits for completion)</td>
</tr>
<tr>
<td><code>bool IsOpen (const)</code></td>
<td>Return true if display is currently open</td>
</tr>
<tr>
<td><code>bool AdapterExists (CoreGraphics::Adapter::Code)</code></td>
<td>Return true if adapter exists (waits for completion)</td>
</tr>
<tr>
<td><code>Util::Array &lt; CoreGraphics::DisplayMode &gt; GetAvailableDisplayModes (CoreGraphics::Adapter::Code, CoreGraphics::PixelFormat::Code)</code></td>
<td>Get available display modes on given adapter (waits for completion)</td>
</tr>
<tr>
<td><code>bool SupportsDisplayMode (CoreGraphics::Adapter::Code, CoreGraphics::DisplayMode)</code></td>
<td>Return true if a given display mode is supported (waits for completion)</td>
</tr>
<tr>
<td><code>CoreGraphics::DisplayMode GetCurrentAdapterDisplayMode (CoreGraphics::Adapter::Code)</code></td>
<td>Get current adapter display mode (i.e. the desktop display mode) (waits for completion)</td>
</tr>
<tr>
<td><code>CoreGraphics::AdapterInfo GetAdapterInfo (CoreGraphics::Adapter::Code)</code></td>
<td>Get general info about display adapter (waits for completion)</td>
</tr>
<tr>
<td><code>void AttachDisplayEventHandler (const Ptr&lt;CoreGraphics::ThreadSafeDisplayEventHandler&gt; &amp;displayEventHandler)</code></td>
<td>Attach a display event handler to the display (waits for completion)</td>
</tr>
</tbody>
</table>
attach a display event handler

```cpp
void RemoveDisplayEventHandler (const Ptr<CoreGraphics::ThreadSafeDisplayEventHandler> &displayEventHandler)
```

remove a display event handler

```cpp
void AttachRenderEventHandler (const Ptr<CoreGraphics::ThreadSafeRenderEventHandler> &renderEventHandler)
```

attach a render event handler

```cpp
void RemoveRenderEventHandler (const Ptr<CoreGraphics::ThreadSafeRenderEventHandler> &renderEventHandler)
```

remove a render event handler

```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
```
<table>
<thead>
<tr>
<th><strong>GetClassFourCC</strong> ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<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>
**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.
AsyncGraphics::GraphicsEntityProxy
AsyncGraphics::GraphicsEntityProxy
Class Reference

#include <graphicsentityproxy.h>

Inheritance diagram for AsyncGraphics::GraphicsEntityProxy:

- Core::RefCounted
- AsyncGraphics::GraphicsEntityProxy
- AsyncGraphics::CameraEntityProxy
Detailed Description


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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GraphicsEntityProxy</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~GraphicsEntityProxy</strong> ()</td>
<td>destructor</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><strong>Graphics::GraphicsEntity::Type</strong> <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>bool <strong>isVisible</strong> () const</td>
<td>return true if entity is set to visible</td>
</tr>
<tr>
<td>const Ptr&lt; StageProxy &gt; &amp; <strong>GetStageProxy</strong> () const</td>
<td>get the stage proxy this entity is attached to</td>
</tr>
<tr>
<td>bool <strong>IsAttachedToStageProxy</strong> () const</td>
<td>return true if entity is attached to stage</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</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></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>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>SetType</strong> <em>(Graphics::GraphicsEntity::Type t)</em></td>
<td>set graphics entity type, called from constructor of subclass</td>
</tr>
<tr>
<td>virtual</td>
<td><strong>Setup</strong> <em>(const Ptr&lt; StageProxy &gt; &amp;stage)</em></td>
<td>called by stage when entity should setup itself</td>
</tr>
<tr>
<td>virtual</td>
<td><strong>Discard</strong> ()</td>
<td>called by stage when entity should discard itself</td>
</tr>
<tr>
<td>void</td>
<td><strong>ValidateEntityHandle</strong> <em>(bool wait=false)</em></td>
<td>validate entity handle</td>
</tr>
<tr>
<td>void</td>
<td><strong>SendCreateMsg</strong> <em>(const Ptr&lt; CreateGraphicsEntity &gt; &amp;msg)</em></td>
<td>send off a specific create message from the subclass</td>
</tr>
<tr>
<td>void</td>
<td><strong>SendMsg</strong> <em>(const Ptr&lt; GraphicsEntityMsg &gt; &amp;msg)</em></td>
<td>send a message to the server-side graphics entity</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsEntityHandleValid</strong> () const</td>
<td>test if the entity handle is valid</td>
</tr>
<tr>
<td>virtual</td>
<td><strong>OnTransformChanged</strong> ()</td>
<td>called when transform matrix changed</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
class AsyncGraphics:
    class GraphicsEntityProxy:
        void Setup(const Ptr<StageProxy> &stageProxy_)
            [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 AsyncGraphics::CameraEntityProxy.

```cpp
class AsyncGraphics:
    class GraphicsEntityProxy:
        void Discard()
            [protected, virtual]

called by stage when entity should discard itself

Discard the server-side graphics entity. This method is called from StageProxy::RemoveEntityProxy(). If the server-side entity hasn’t been created yet, this method needs to wait for completion, since the entity handle won’t be available yet!

```cpp
class AsyncGraphics:
    class GraphicsEntityProxy:
        void ValidateEntityHandle(bool wait = false)
            [protected]

validate entity handle

Tries to validate the entity handle by checking whether the pending creation message has already been handled by the server side. If the wait flag is true, the method will wait until the entity has been created on the server side. The default is to NOT wait. If the graphics entity handle becomes valid, all pending messages will be sent off to the

```cpp
class AsyncGraphics:
    class GraphicsEntityProxy:
        void SendCreateMsg(const Ptr<CreateGraphicsEntity> &msg)
            [protected]

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 AsyncGraphics::GraphicsEntityProxy::SendMsg(
    const Ptr<GraphicsEntityMsg> msg ) [protected]
```

send a message to the server-side graphics entity

Send a generic GraphicsEntityMsg to the server-side entity. Do not initialize the entity handle of the message before calling this method, since the server-side entity doesn't have to exist yet. In this case, the message will be queued up and sent off as soon as the server-side entity becomes valid.

```cpp
void AsyncGraphics::GraphicsEntityProxy::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 AsyncGraphics::CameraEntityProxy.

```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.
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.
AsyncGraphics::GraphicsServerProxy
AsyncGraphics::GraphicsServerProxy
Class Reference

#include <graphicsserverproxy.h>

Inheritance diagram for AsyncGraphics::GraphicsServerProxy:
Detailed Description

Client-side proxy of the GraphicsServer. Used to create and update StageProxies and ViewProxies.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GraphicsServerProxy ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~GraphicsServerProxy ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Open ()</strong></td>
<td>open the graphics server proxy</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close the graphics server proxy</td>
</tr>
<tr>
<td><strong>bool IsOpen () const</strong></td>
<td>return true if graphics server is open</td>
</tr>
<tr>
<td><strong>void OnFrame ()</strong></td>
<td>perform per-frame updates</td>
</tr>
<tr>
<td><strong>Ptr&lt; StageProxy &gt; CreateStageProxy (const Util::StringAtom &amp;name, const Core::Rtti &amp;stageBuilderClass, const Attr::AttributeContainer &amp;stageBuilderAttrs)</strong></td>
<td>create a stage proxy</td>
</tr>
<tr>
<td><strong>void DiscardStageProxy (const Ptr&lt; StageProxy &gt; &amp;stageProxy)</strong></td>
<td>discard a stage proxy object</td>
</tr>
<tr>
<td><strong>void DiscardAllStageProxies ()</strong></td>
<td>discard all stage proxy objects</td>
</tr>
<tr>
<td><strong>bool HasStageProxy (const Util::StringAtom &amp;name) const</strong></td>
<td>return true if a stage proxy exists by name</td>
</tr>
<tr>
<td><strong>const Ptr&lt; StageProxy &gt; &amp; GetStageProxyByName (const Util::StringAtom &amp;name) const</strong></td>
<td>lookup a stage proxy by name</td>
</tr>
<tr>
<td><strong>const Util::Array &lt; Ptr&lt; StageProxy &gt; &gt; &amp; GetStageProxies () const</strong></td>
<td>get all stage proxies</td>
</tr>
<tr>
<td><strong>Ptr&lt; ViewProxy &gt; CreateViewProxy (const Core::Rtti &amp;viewClass, const Util::StringAtom &amp;viewName, const Util::StringAtom</strong></td>
<td></td>
</tr>
</tbody>
</table>
&stageName, const Resources::ResourceId &frameShaderName, bool isDefaultView=false)
create a view proxy

void DiscardViewProxy (const Ptr<ViewProxy> &view)
discard a view proxy

void DiscardAllViewProxies ()
discard all view proxies

bool HasViewProxy (const Util::StringAtom &name) const
return true if a view proxy exists by name

const Ptr<ViewProxy> & GetViewProxyByName (const Util::StringAtom &name) const
lookup a view proxy by name

const Util::Array<Ptr<ViewProxy>> & GetViewProxies () const
get all view proxies

const Ptr<ViewProxy> & GetDefaultViewProxy () const
get the default view proxy

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
<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>
<tr>
<td>const <code>Util::String &amp;</code></td>
<td><code>GetClassName () const</code></td>
</tr>
<tr>
<td></td>
<td><code>get the class name</code></td>
</tr>
<tr>
<td><code>Util::FourCC</code></td>
<td><code>GetClassFourCC () const</code></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>Type</th>
<th>Function</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 (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.
AsyncGraphics::Handle
AsyncGraphics::Handle Class Reference

#include <handle.h>
Detailed Description

A private **Handle** typedef to safely pass data between a server and a client thread. Don't assume anything about the content of the Handle!

(C) 2007 Radon Labs GmbH
AsyncGraphics::ViewProxy
AsyncGraphics::ViewProxy Class Reference

#include <viewproxy.h>

Inheritance diagram for AsyncGraphics::ViewProxy:

```
AsyncGraphics::ViewProxy
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
| Core::RefCounted
```
Detailed Description

A client-side proxy of a `Graphics::View` in the AsyncGraphic subsystem.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ViewProxy</strong> ()</td>
<td>virtual constructor</td>
</tr>
<tr>
<td>virtual <strong>~ViewProxy</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td>bool <strong>IsValid</strong> () const</td>
<td>return true if the <strong>ViewProxy</strong> is valid</td>
</tr>
<tr>
<td>const <strong>Util::StringAtom</strong> &amp; <strong>GetName</strong> () const</td>
<td>get name of view-proxy</td>
</tr>
<tr>
<td>const <strong>Core::Rtti</strong> &amp; <strong>GetViewClass</strong> () const</td>
<td>get the class-type of the server-side view object</td>
</tr>
<tr>
<td>const <strong>Util::StringAtom</strong> &amp; <strong>GetStageName</strong> () const</td>
<td>get the name of the stage this view is attached to</td>
</tr>
<tr>
<td>const <strong>Resources::ResourceId</strong> &amp; <strong>GetFrameShaderName</strong> () const</td>
<td>get the name of the frame-shader this view will use for rendering</td>
</tr>
<tr>
<td>bool <strong>IsDefaultView</strong> () const</td>
<td>check whether this is the default view</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>Class</td>
<td>Method</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td><code>class</code></td>
<td><code>bool IsA (const Util::String &amp;rttiName) const</code></td>
</tr>
<tr>
<td></td>
<td><code>bool IsA (const Util::FourCC &amp;rttiFourCC) const</code></td>
</tr>
<tr>
<td></td>
<td><code>const Util::String &amp; 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</th>
<th>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBUA3_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.
AsyncHttp::AsyncHttpHandler
AsyncHttp::AsyncHttpGet Handler Class Reference

#include <asynchtphpandler.h>
Detailed Description

Runs the AsyncHttp thread, and processes messages to theHttp subsystem.

(C) 2007 Radon Labs GmbH
AsyncHttp::AsyncHttpInterface
AsyncHttp::AsyncHttpInterface Class Reference

#include <asynchtppinterface.h>

Inheritance diagram for AsyncHttp::AsyncHttpInterface:
Detailed Description

Central interface object of the asynchronous Http subsystem.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsyncHttpInterface()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~AsyncHttpInterface()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void Open()</td>
<td>Open the interface object</td>
</tr>
<tr>
<td>void SetName(const Util::String &amp;n)</td>
<td>Set the name of the async port (required)</td>
</tr>
<tr>
<td>const Util::String &amp; GetName()</td>
<td>Get the name of the async port</td>
</tr>
<tr>
<td>void SetThreadPriority(const Threading::Thread::Priority pri)</td>
<td>Set optional thread priority</td>
</tr>
<tr>
<td>Threading::Thread::Priority GetThreadPriority() const</td>
<td>Get optional thread priority</td>
</tr>
<tr>
<td>void SetThreadStackSize(unsigned int s)</td>
<td>Set optional thread stack size</td>
</tr>
<tr>
<td>unsigned int GetThreadStackSize() const</td>
<td>Get optional thread stack size</td>
</tr>
<tr>
<td>void SetWaitForMessages(bool b)</td>
<td>Wait-for-messages or run-continously? (default is wait-for-message)</td>
</tr>
<tr>
<td>bool GetWaitForMessages() const</td>
<td>Get wait-for-message mode</td>
</tr>
<tr>
<td>void AttachHandler(const Ptr&lt;Handler&gt; &amp;h)</td>
<td>Attach a handler to the port (call before open!)</td>
</tr>
<tr>
<td>virtual void Close()</td>
<td>Close the async port</td>
</tr>
<tr>
<td>bool isOpen() const</td>
<td>Return true if port is open</td>
</tr>
<tr>
<td>void Send(const Ptr&lt;Message&gt; &amp;msg)</td>
<td>Send an asynchronous message to the port</td>
</tr>
<tr>
<td>void SendWait(const Ptr&lt;Message&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><code>Send()</code></td>
<td>send a message and wait for completion</td>
</tr>
<tr>
<td><code>Wait(const Ptr&lt; Message &gt; &amp;msg)</code></td>
<td>wait for a message to be handled</td>
</tr>
<tr>
<td><code>Peek(const Ptr&lt; Message &gt; &amp;msg)</code></td>
<td>peek a message whether it has been handled</td>
</tr>
<tr>
<td><code>Cancel(const Ptr&lt; Message &gt; &amp;msg)</code></td>
<td>cancel a pending message</td>
</tr>
<tr>
<td><code>Flush()</code></td>
<td>wait until all pending messages have been handled</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, 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

```c
static void DumpRefCountingLeaks ()

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

virtual void OnCreateHandlers ()

derive in subclass, create and attach handlers from here
Member Function Documentation

```cpp
void Messaging::AsyncPort::AttachHandler(
    const Ptr<Handler> &h) [inherited]
```

attach a handler to the port (call before open!)

Called by **OnCreateHandlers()** method of subclass to attach a handler to the port.

```cpp
void Messaging::AsyncPort::Close()
```

close the async port

Closes the async port.

```cpp
void Messaging::AsyncPort::Send(
    const Ptr<Message> &msg) [inherited]
```

send an asynchronous message to the port

Handle an asynchronous message and return immediately. If the caller expects any results from the message he can poll with the **AsyncPort::Peek()** method, or he may wait for the message to be handled with the **AsyncPort::Wait()** method.

```cpp
void Messaging::AsyncPort::SendWait(
    const Ptr<Message> &msg) [inherited]
```

send a message and wait for completion

Send an asynchronous message and wait until the message has been handled.
void Messaging::AsyncPort::Wait(const Ptr<Message> &msg) [inherited]

wait for a message to be handled

This method will wait until a message has been handled. If the caller expects any return arguments from the message handling it can use this method to wait for the results.

bool Messaging::AsyncPort::Peek(const Ptr<Message> &msg) [inherited]

peek a message whether it has been handled

This method peeks whether a message has already been handled. If the caller expects any return arguments from the message handling it can use this message to check whether the results are ready using this non-blocking method. The caller can also wait for the results to become ready using the \texttt{Wait()} method.

void Messaging::AsyncPort::Cancel(const Ptr<Message> &msg) [inherited]

cancel a pending message

This method will cancel a pending message.

void Messaging::AsyncPort::Flush() [inherited]

wait until all pending messages have been handled

This method will wait until ALL pending messages have been handled. Note that this method will be called automatically before the \texttt{AsyncPort} shutdown.

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.
AsyncHttp::HttpServerProxy
AsyncHttp::HttpServerProxy Class Reference

#include <httpserverproxy.h>
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
Attr::AccessMode
Attr::AccessMode Class Reference

#include <accessmode.h>
Detailed Description

(C) 2006 Radon Labs GmbH
Attr::Attribute
Attr::Attribute Class Reference

#include <attribute.h>

Inheritance diagram for Attr::Attribute:

```
Util::KeyValuePair< KEYTYPE, VALUETYPE >

Attr::Attribute

< KEYTYPE, VALUETYPE >" shape="rect" coords="0,0,276,24">
```
Detailed Description

A compiletime-typesafe key/value pair.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Attribute ()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>Attribute (const Attribute &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>Attribute (const AttrId &amp;id)</code></td>
<td>construct from typeless attribute id</td>
</tr>
<tr>
<td><code>Attribute (const BoolAttrId &amp;id, bool val)</code></td>
<td>construct from bool</td>
</tr>
<tr>
<td><code>Attribute (const FloatAttrId &amp;id, float val)</code></td>
<td>construct from float</td>
</tr>
<tr>
<td><code>Attribute (const IntAttrId &amp;id, int val)</code></td>
<td>construct from int</td>
</tr>
<tr>
<td><code>Attribute (const Matrix44AttrId &amp;id, const Math::matrix44 &amp;val)</code></td>
<td>construct from matrix44</td>
</tr>
<tr>
<td><code>Attribute (const StringAttrId &amp;id, const Util::String &amp;val)</code></td>
<td>construct from string</td>
</tr>
<tr>
<td><code>Attribute (const Float4AttrId &amp;id, const Math::float4 &amp;val)</code></td>
<td>construct from float4</td>
</tr>
<tr>
<td><code>Attribute (const GuidAttrId &amp;id, const Util::Guid &amp;val)</code></td>
<td>construct from guid</td>
</tr>
<tr>
<td><code>Attribute (const BlobAttrId &amp;id, const Util::Blob &amp;val)</code></td>
<td>construct from blob</td>
</tr>
</tbody>
</table>

```cpp
void SetAttrId (const AttrId &id) const
    set attribute id
```

```cpp
const AttrId & GetAttrId () const
    get attribute id
```

```cpp
const Util::String & GetName () const
    get name of attribute
```
const Util::FourCC & GetFourCC () const
generate fourcc of attribute

ValueType GetValueType () const
get value type of attribute

AccessMode GetAccessMode () const
generate access mode of attribute

void Clear ()
clear the attribute's value

void operator= (const Attribute &rhs)
assignment operator

void operator= (bool rhs)
bool assignment operator

void operator= (float rhs)
float assignment operator

void operator= (int rhs)
int assignment operator

void operator= (const Math::matrix44 &rhs)
matrix44 assignment operator

void operator= (const Util::String &rhs)
string assignment operator

void operator= (const Math::float4 &rhs)
float4 assignment operator

void operator= (const Util::Guid &rhs)
guid assignment operator

void operator= (const Util::Blob &rhs)
blob assignment operator

bool operator== (const Attribute &rhs) const
equality operator

bool operator== (const Util::String &rhs) const
string equality operator

bool operator== (int rhs) const
int equality operator

bool operator== (float rhs) const
float equality operator

bool operator== (bool rhs) const
bool equality operator
| bool operator== (const Math::float4 &rhs) const |
| float4 equality operator |
| bool operator== (const Util::Guid &rhs) const |
| guid equality operator |
| bool operator!=(const Attribute &rhs) const |
| equality operator |
| bool operator!=(const Util::String &rhs) const |
| string equality operator |
| bool operator!=(int rhs) const |
| int equality operator |
| bool operator!=(float rhs) const |
| float equality operator |
| bool operator!=(bool rhs) const |
| bool equality operator |
| bool operator!=(const Math::float4 &rhs) const |
| float4 equality operator |
| bool operator!=(const Util::Guid &rhs) const |
| guid equality operator |
| void SetBool (bool val) |
| set bool content |
| bool GetBool () const |
| get bool content |
| void SetInt (int val) |
| set int content |
| int GetInt () const |
| get int content |
| void SetFloat (float val) |
| set float content |
| float GetFloat () const |
| get float content |
| void SetString (const Util::String &val) |
| set string content |
| const Util::String & GetString () const |
| get string content |
| void SetMatrix44 (const Math::matrix44 &val) |
| set matrix44 value |
const Math::matrix44 & GetMatrix44 () const
get matrix44 value

void SetFloat4 (const Math::float4 &val)
set float4 value

Math::float4 GetFloat4 () const
get float4 value

void SetGuid (const Util::Guid &val)
set guid value

const Util::Guid & GetGuid () const
get guid value

void SetBlob (const Util::Blob &val)
set blob value

const Util::Blob & GetBlob () const
get blob value

void SetValue (const Util::Variant &val)
set generic value

const Util::Variant & GetValue () const
get generic value

void SetValueFromString (const Util::String &str)
set value from string (convert as necessary)

Util::String ValueAsString () const
convert the content to a string

bool operator== (const KeyValuePair< KEYTYPE, VALUETYPE > &rhs) const
equality operator

bool operator!= (const KeyValuePair< KEYTYPE, VALUETYPE > &rhs) const
inequality operator

bool operator> (const KeyValuePair< KEYTYPE, VALUETYPE > &rhs) const
greater operator

bool operator>> (const KeyValuePair< KEYTYPE, VALUETYPE > &rhs) const
greater-or-equal operator

bool operator< (const KeyValuePair< KEYTYPE, VALUETYPE > &rhs) const
lesser operator
| bool operator<= (const KeyValuePair< KEYTYPE, VALUETYPE > &rhs) const
| \( \text{lesser-or-equal operator} \) |
| VALUETYPE & Value () \( \text{read/write access to value} \) |
| const VALUETYPE & Value () const \( \text{read access to key} \) |
| const KEYTYPE & Key () const \( \text{read access to key} \) |
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Static Value Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StringToValueType</code></td>
<td>(const <code>Util::String</code> &amp;s)</td>
<td>convert string to type</td>
</tr>
<tr>
<td><code>Util::String</code></td>
<td><code>ValueToString</code></td>
<td>convert type to string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Tue Feb 19 12:16:41 2008
Attr::AttributeContainer
Attr::AttributeContainer Class Reference

#include <attributecontainer.h>
Detailed Description

A simple container for attributes.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AttributeContainer()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~AttributeContainer()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>bool HasAttr(const AttrId &amp;attrId)</code> const</td>
<td>Check if an attribute exists in the container</td>
</tr>
<tr>
<td><code>void SetAttr(const Attribute &amp;attr)</code></td>
<td>Set a single attribute, new or existing</td>
</tr>
<tr>
<td><code>const Attribute &amp; GetAttr(const AttrId &amp;attrId)</code> const</td>
<td>Get a single attribute</td>
</tr>
<tr>
<td><code>const Util::Dictionary&lt;AttrId, Attribute&gt; &amp; GetAttrs()</code> const</td>
<td>Read access to the attribute array</td>
</tr>
<tr>
<td><code>void Clear()</code></td>
<td>Clear the attribute container</td>
</tr>
<tr>
<td><code>void AddAttr(const Attribute &amp;attr)</code></td>
<td>Add a new attribute, faster than <code>SetAttr()</code>, but attribute may not exist!</td>
</tr>
<tr>
<td><code>void SetBool(const BoolAttrId &amp;attrId, bool val)</code></td>
<td>Set bool value</td>
</tr>
<tr>
<td><code>bool GetBool(const BoolAttrId &amp;attrId)</code> const</td>
<td>Get bool value</td>
</tr>
<tr>
<td><code>bool GetBool(const BoolAttrId &amp;attrId, bool defaultValue)</code> const</td>
<td>Get bool value with default if not exists</td>
</tr>
<tr>
<td><code>void SetFloat(const FloatAttrId &amp;attrId, float val)</code></td>
<td>Set float value</td>
</tr>
<tr>
<td><code>float GetFloat(const FloatAttrId &amp;attrId)</code> const</td>
<td>Get float value</td>
</tr>
<tr>
<td><code>float GetFloat(const FloatAttrId &amp;attrId, float defaultValue)</code> const</td>
<td>Get float value with default if not exists</td>
</tr>
</tbody>
</table>
| `void SetInt(const IntAttrId &attrId, int val)` | }
set int value

int GetInt (const IntAttrId &attrId) const
get int value

int GetInt (const IntAttrId &attrId, int defaultValue) const
get int value with default if not exists

const Util::String & GetString (const StringAttrId &attrId) const
get string value

const Util::String & GetString (const StringAttrId &attrId, const Util::String &defaultValue) const
get string value with default if not exists

void SetString (const StringAttrId &attrId, const Util::String &val)
set string value

const Util::String & GetString (const StringAttrId &attrId) const
get string value

const Util::String & GetString (const StringAttrId &attrId, const Util::String &defaultValue) const
get string value with default if not exists

void SetFloat4 (const Float4AttrId &attrId, const Math::float4 &val)
set float4 value

Math::float4 GetFloat4 (const Float4AttrId &attrId) const
get float4 value

Math::float4 GetFloat4 (const Float4AttrId &attrId, const Math::float4 &defaultValue) const
get float4 value with default if not exists

void SetMatrix44 (const Matrix44AttrId &attrId, const Math::matrix44 &val)
set matrix44 value

const Math::matrix44 & GetMatrix44 (const Matrix44AttrId &attrId) const
get matrix44 value

const Math::matrix44 & GetMatrix44 (const Matrix44AttrId &attrId, const Math::matrix44 &defaultValue) const
get matrix44 value with default if not exists

void SetGuid (const GuidAttrId &attrId, const Util::Guid &guid)
set guid value

const Util::Guid & GetGuid (const GuidAttrId &attrId) const
get guid value

const Util::Guid & GetGuid (const GuidAttrId &attrId, const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Util::Guid</code> &amp;defaultValue) const</td>
<td>get guid value with default if not exists</td>
</tr>
<tr>
<td><code>void</code> SetBlob (const <code>BlobAttrId</code> &amp;attrId, const <code>Util::Blob</code> &amp;blob)</td>
<td>set blob value</td>
</tr>
<tr>
<td><code>const Util::Blob</code> &amp; GetBlob (const <code>BlobAttrId</code> &amp;attrId) const</td>
<td>get blob value</td>
</tr>
<tr>
<td><code>const Util::Blob</code> &amp; GetBlob (const <code>BlobAttrId</code> &amp;attrId, const <code>Util::Blob</code> &amp;default) const</td>
<td>get blob value with default if not exists</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool Attr::AttributeContainer::HasAttr (AttrId attrId) const
check if an attribute exists in the container
Return true if an attribute exists.

void Attr::AttributeContainer::SetAttr (Attribute attr) &
set a single attribute, new or existing
Set a generic attribute. If the attribute exists, its value will be overwritten and a type check will be made (you can't overwrite an attribute with a different types one). If the attribute doesn't exist, a new attribute will be created.

const Attribute & Attr::AttributeContainer::GetAttr (AttrId attrId) const
get a single attribute
Get generic attribute. Throws a hard error if the attribute doesn't exist.

const Util::Dictionary<AttrId, Attribute> & Attr::AttributeContainer::GetAttrs
read access to the attribute array
This method provides direct read access to the attributes.

void Attr::AttributeContainer::Clear ()
clear the attribute container
This method clears the attributes in the attribute container.

```cpp
void Attr::AttributeContainer::AddAttr
    ( const Attribute &attr )
```

add a new attribute, faster then `SetAttr()`, but attribute may not exist!

Add a new attribute. The attribute may not exist yet in the container, otherwise the result is undefined. This is faster then `SetAttr()` when many attributes are added.
Attr::AttributeDefinition
Attr::AttributeDefinition< VALUETYPE, TYPE > Class Template Reference

#include <attributedefinition.h>

Inheritance diagram for Attr::AttributeDefinition< VALUETYPE, TYPE >:

```
Attr::AttributeDefinitionBase

Attr::AttributeDefinition< VALUETYPE, TYPE >
```

< VALUETYPE, TYPE >._map" border="0" alt="">
< VALUETYPE, TYPE >._map"
Detailed Description

template<class VALUETYPE, class TYPE>
class Attr::AttributeDefinition< VALUETYPE, TYPE >

This extends the typeless AttributeDefinitionBase class by a typed template class, which adds compiletime-type-safety to attribute definitions.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AttributeDefinition</code> (const <code>Util::String</code> &amp;n, const <code>Util::FourCC</code> &amp;fourCC, <code>AccessMode</code> am, TYPE defVal)</td>
<td>constructor</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp; <code>GetName</code> () const</td>
<td>get attribute name</td>
</tr>
<tr>
<td>const <code>Util::FourCC</code> &amp; <code>GetFourCC</code> () const</td>
<td>get fourcc code</td>
</tr>
<tr>
<td>bool <code>IsDynamic</code> () const</td>
<td>return true if this is a dynamic attribute</td>
</tr>
<tr>
<td>const <code>Util::Variant</code> &amp; <code>GetDefaultValue</code> () const</td>
<td>get default value</td>
</tr>
<tr>
<td><code>AccessMode</code> <code>GetAccessMode</code> () const</td>
<td>get access type</td>
</tr>
<tr>
<td><code>ValueType</code> <code>GetValueType</code> () const</td>
<td>get value type</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static void Destroy ()</strong></td>
<td>Static destruction method - call to cleanup the registry hashtable</td>
</tr>
<tr>
<td><strong>static const AttributeDefinitionBase * FindByName (const Util::String &amp;n)</strong></td>
<td>Find by name</td>
</tr>
<tr>
<td><strong>static const AttributeDefinitionBase * FindByFourCC (const Util::FourCC &amp;fcc)</strong></td>
<td>Find by FourCC</td>
</tr>
<tr>
<td><strong>static void RegisterDynamicAttribute (const Util::String &amp;name, const Util::FourCC &amp;fourCC, ValueType valueType, AccessMode accessMode)</strong></td>
<td>Register a dynamic attribute</td>
</tr>
<tr>
<td><strong>static void ClearDynamicAttributes ()</strong></td>
<td>Clear all dynamic attributes</td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>Register ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>register an attribute definition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>Unregister ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>unregister an attribute definition</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Attr::AttributeDefinitionBase::Destroy() [static, inherited]

static destruction method - call to cleanup the registry hashtable

Cleanup the name registry!

void Attr::AttributeDefinitionBase::RegisterDynamicAttribute(const Util::String &name,
                                                            const Util::FourCC &fourCC,
                                                            Attr::ValueType valueType,
                                                            Attr::AccessMode accessMode) [static, inherited]

register a dynamic attribute

This method registers a new dynamic attribut.

void Attr::AttributeDefinitionBase::ClearDynamicAttributes() [static, inherited]

clear all dynamic attributes

This clears all dynamic attributes.

void Attr::AttributeDefinitionBase::Register() [protected, inherited]

register an attribute definition

Register this static attribute definition in the name and fourcc registries. Since the order of initialization is not defined for static objects we need to use pointers and creation-on-demand for the registry objects.

void Attr::AttributeDefinitionBase::Unregister() [protected, inherited]
unregister an attribute definition

Unregister this attribute definition from the registries.

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:41 2008
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Attr::AttributeDefinitionBase
Attr::AttributeDefinitionBase Class Reference

#include <attributedefinitionbase.h>

Inheritance diagram for Attr::AttributeDefinitionBase:

```
< VALUETYPE, TYPE >" shape="rect" coords="0,56,276,80">
```

Detailed Description

Implements a universal attribute definition, consisting of an attribute name, attribute fourcc code, value type and access type.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AttributeDefinitionBase</code></td>
<td>(const <code>Util::String</code> &amp;name, const <code>Util::FourCC</code> &amp;fourCC, <code>AccessMode</code> accessMode, bool isDynamic)</td>
</tr>
<tr>
<td><code>constructor</code></td>
<td></td>
</tr>
<tr>
<td><code>AttributeDefinitionBase</code></td>
<td>(const <code>Util::String</code> &amp;name, const <code>Util::FourCC</code> &amp;fourCC, <code>AccessMode</code> accessMode, bool defVal, bool isDynamic)</td>
</tr>
<tr>
<td><code>bool constructor</code></td>
<td></td>
</tr>
<tr>
<td><code>AttributeDefinitionBase</code></td>
<td>(const <code>Util::String</code> &amp;name, const <code>Util::FourCC</code> &amp;fourCC, <code>AccessMode</code> accessMode, int defVal, bool isDynamic)</td>
</tr>
<tr>
<td><code>int constructor</code></td>
<td></td>
</tr>
<tr>
<td><code>AttributeDefinitionBase</code></td>
<td>(const <code>Util::String</code> &amp;name, const <code>Util::FourCC</code> &amp;fourCC, <code>AccessMode</code> accessMode, float defVal, bool isDynamic)</td>
</tr>
<tr>
<td><code>float constructor</code></td>
<td></td>
</tr>
<tr>
<td><code>AttributeDefinitionBase</code></td>
<td>(const <code>Util::String</code> &amp;name, const <code>Util::FourCC</code> &amp;fourCC, <code>AccessMode</code> accessMode, const <code>Util::String</code> &amp;defVal, bool isDynamic)</td>
</tr>
<tr>
<td><code>string constructor</code></td>
<td></td>
</tr>
<tr>
<td><code>AttributeDefinitionBase</code></td>
<td>(const <code>Util::String</code> &amp;name, const <code>Util::FourCC</code> &amp;fourCC, <code>AccessMode</code> accessMode, const <code>Math::float4</code> &amp;defVal, bool isDynamic)</td>
</tr>
<tr>
<td><code>float4 constructor</code></td>
<td></td>
</tr>
<tr>
<td><code>AttributeDefinitionBase</code></td>
<td>(const <code>Util::String</code> &amp;name, const <code>Util::FourCC</code> &amp;fourCC, <code>AccessMode</code> accessMode, const <code>Math::matrix44</code> &amp;defVal, bool isDynamic)</td>
</tr>
<tr>
<td><code>matrix44 constructor</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>AttributeDefinitionBase (const Util::String &amp;name, const Util::FourCC &amp;fourCC, AccessMode accessMode, const Util::Blob &amp;defVal, bool isDynamic)</code></td>
<td>blob constructor</td>
</tr>
<tr>
<td><code>AttributeDefinitionBase (const Util::String &amp;name, const Util::FourCC &amp;fourCC, AccessMode accessMode, const Util::Guid &amp;defVal, bool isDynamic)</code></td>
<td>guid constructor</td>
</tr>
<tr>
<td><code>~AttributeDefinitionBase ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetName () const</code></td>
<td>get attribute name</td>
</tr>
<tr>
<td><code>const Util::FourCC &amp; GetFourCC () const</code></td>
<td>get fourcc code</td>
</tr>
<tr>
<td><code>bool IsDynamic () const</code></td>
<td>return true if this is a dynamic attribute</td>
</tr>
<tr>
<td><code>const Util::Variant &amp; GetDefaultValue () const</code></td>
<td>get default value</td>
</tr>
<tr>
<td><code>AccessMode GetAccessMode () const</code></td>
<td>get access type</td>
</tr>
<tr>
<td><code>ValueType GetValue Type () const</code></td>
<td>get value type</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>Destroy</strong> ()</td>
<td>static destruction method - call to cleanup the registry hashtable</td>
</tr>
<tr>
<td>static const AttributeDefinitionBase * <strong>FindByName</strong> (const Util::String &amp;n)</td>
<td>find by name</td>
</tr>
<tr>
<td>static const AttributeDefinitionBase * <strong>FindByFourCC</strong> (const Util::FourCC &amp;fcc)</td>
<td>find by FourCC</td>
</tr>
<tr>
<td>static void <strong>RegisterDynamicAttribute</strong> (const Util::String &amp;name, const Util::FourCC &amp;fourCC, ValueType valueType, AccessMode accessMode)</td>
<td>register a dynamic attribute</td>
</tr>
<tr>
<td>static void <strong>ClearDynamicAttributes</strong> ()</td>
<td>clear all dynamic attributes</td>
</tr>
</tbody>
</table>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Description</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>register an attribute definition</td>
<td><code>Register()</code></td>
<td></td>
</tr>
<tr>
<td>unregister an attribute definition</td>
<td><code>Unregister()</code></td>
<td></td>
</tr>
</tbody>
</table>
**Member Function Documentation**

void
Attr::AttributeDefinitionBase::Destroy ( ) [static]

static destruction method - call to cleanup the registry hashtable

Cleanup the name registry!

void
Attr::AttributeDefinitionBase::RegisterDynamicAttribute (const Util::String & name,
const Util::FourCC & fourCC,
Attr::ValueType valueType,
Attr::AccessMode accessMode ) [static]

register a dynamic attribute

This method registers a new dynamic attribut.

void
Attr::AttributeDefinitionBase::ClearDynamicAttributes ( ) [static]

clear all dynamic attributes

This clears all dynamic attributes.

void
Attr::AttributeDefinitionBase::Register ( ) [protected]

register an attribute definition

Register this static attribute definition in the name and fourcc registries. Since the order of initialization is not defined for static objects we need to use pointers and creation-on-demand for the registry objects.

void
Attr::AttributeDefinitionBase::Unregister ( ) [protected]
unregister an attribute definition

Unregister this attribute definition from the registries.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Attr::AttributeTable
#include <attributetable.h>

Inheritance diagram for Attr::AttributeTable:
Detailed Description

A table of attributes with a compact memory footprint and fast random access. Table columns are defined by attribute ids which associate a name, a fourcc code, a datatype and an access mode (ReadWrite, ReadOnly) to the table. Attribute values are stored in one big chunk of memory without additional overhead. Table cells can have the NULL status, which means the cell contains no value.

The table's value buffer consists of 4-byte aligned rows, each row consists of a bitfield with 2 bits per row (one bit is set if a column/row value is valid, the other is used as modified-marker).

The header-bitfield is padded to 4-byte. After the header field follow the value fields, one field for each column. The size of the field depends on the datatype of the column, the minimum field size of 4 bytes for data alignment reasons:

- Bool: `sizeof(int)` usually 4 bytes
- Int: `sizeof(int)` usually 4 bytes
- Float: `sizeof(float)` usually 4 bytes
- Float4: `sizeof(float4)` usually 16 bytes
- Matrix44: `sizeof(matrix44)` usually 48 bytes
- String: `sizeof(char*)` usually 4 bytes

The AttributeTable object keeps track of all changes (added columns, added rows, modified rows, modified values).

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AttributeTable ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~AttributeTable ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void SetModifiedTracking (bool b)</strong></td>
<td>enable/disable modified tracking (default is on)</td>
</tr>
<tr>
<td><strong>bool GetModifiedTracking () const</strong></td>
<td>get modified tracking flag</td>
</tr>
<tr>
<td><strong>bool IsModified () const</strong></td>
<td>return true if the object has been modified since the last</td>
</tr>
<tr>
<td></td>
<td>ResetModifiedState()</td>
</tr>
<tr>
<td><strong>void ResetModifiedState ()</strong></td>
<td>reset all the modified bits in the table</td>
</tr>
<tr>
<td><strong>void Clear ()</strong></td>
<td>clear the table object</td>
</tr>
<tr>
<td><strong>void PrintDebug ()</strong></td>
<td>print the contents of the table for debugging reasons</td>
</tr>
<tr>
<td><strong>void BeginAddColumns (bool recordNewColumns=true)</strong></td>
<td>optional: call before adding columns, speeds up adding many</td>
</tr>
<tr>
<td></td>
<td>columns at once</td>
</tr>
<tr>
<td><strong>void AddColumn (const AttrId &amp;id, bool recordNewColumn=true)</strong></td>
<td>add a column</td>
</tr>
<tr>
<td><strong>void EndAddColumns ()</strong></td>
<td>optional: call after adding columns, speeds up adding many</td>
</tr>
<tr>
<td></td>
<td>columns at once</td>
</tr>
<tr>
<td><strong>bool HasColumn (const AttrId &amp;id) const</strong></td>
<td>return true if a column exists</td>
</tr>
<tr>
<td><strong>IndexT GetColumnIndex (const AttrId &amp;id)</strong></td>
<td>return index of column by id</td>
</tr>
<tr>
<td><strong>SizeT GetNumColumns () const</strong></td>
<td>get number of columns</td>
</tr>
<tr>
<td><strong>const AttrId &amp; GetColumnId (IndexT colIndex)</strong></td>
<td>const</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>get column definition at index</td>
<td></td>
</tr>
<tr>
<td>const Util::String &amp; GetColumnName (IndexT colIndex) const</td>
<td>get a column's name</td>
</tr>
<tr>
<td>const Util::FourCC &amp; GetColumnFourCC (IndexT colIndex) const</td>
<td>get column FourCC</td>
</tr>
<tr>
<td>AccessMode GetColumnAccessMode (IndexT colIndex) const</td>
<td>get a column's access mode</td>
</tr>
<tr>
<td>ValueType GetColumnType (IndexT colIndex) const</td>
<td>get a column's value type</td>
</tr>
<tr>
<td>const Util::Array &amp; GetNewColumnIndices () const</td>
<td>return indices of columns added since the last ResetModifiedState()</td>
</tr>
<tr>
<td>const Util::Array &amp; GetReadWriteColumnIndices () const</td>
<td>return indices of all ReadWrite columns</td>
</tr>
<tr>
<td>IndexT AddRow ()</td>
<td>add a row to the table, returns index of new row</td>
</tr>
<tr>
<td>void ClearNewRowFlags ()</td>
<td>clear the new row flags, so that new rows are treated like updated rows</td>
</tr>
<tr>
<td>void ClearDeletedRowsFlags ()</td>
<td>clear deleted rows flags</td>
</tr>
<tr>
<td>void DeleteRow (IndexT rowIndex)</td>
<td>mark a row as deleted from the table</td>
</tr>
<tr>
<td>void DeleteAllRows ()</td>
<td>mark all rows as deleted</td>
</tr>
<tr>
<td>bool IsRowDeleted (IndexT rowIndex) const</td>
<td>return true if row has been marked as deleted</td>
</tr>
<tr>
<td>IndexT CopyRow (IndexT rowIndex)</td>
<td>create a new row as copy of another row</td>
</tr>
<tr>
<td>IndexT CopyExtRow (AttributeTable *other, IndexT otherRowIndex, bool createMissingRows=false)</td>
<td>create a new row as copy of a row from another value table</td>
</tr>
<tr>
<td>SizeT GetNumRows () const</td>
<td>get number of rows in table</td>
</tr>
<tr>
<td>bool HasNewRows () const</td>
<td>return true if table has new rows</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool HasModifiedRows () const</code></td>
<td>return true if table has modified rows</td>
</tr>
<tr>
<td><code>bool HasDeletedRows () const</code></td>
<td>return true if table has deleted rows</td>
</tr>
<tr>
<td><code>bool IsRowModified (IndexT rowIndex) const</code></td>
<td>return true if a row has been modified since the last <code>ResetModifiedState()</code></td>
</tr>
<tr>
<td><code>const Util::Array &lt; IndexT &gt; &amp; GetNewRowIndices () const</code></td>
<td>return indices of rows added since the last <code>ResetModifiedState()</code></td>
</tr>
<tr>
<td><code>const Util::Array &lt; IndexT &gt; &amp; GetDeletedRowIndices () const</code></td>
<td>return indices of rows deleted since the last <code>ResetModifiedState()</code></td>
</tr>
<tr>
<td><code>Util::Array&lt; IndexT &gt; GetModifiedRowsExcludeNewAndDeletedRows () const</code></td>
<td>return array of modified rows, exclude rows marked as rows</td>
</tr>
<tr>
<td><code>void ReserveRows (SizeT numRows)</code></td>
<td>reserve rows to reduce re-allocation overhead</td>
</tr>
<tr>
<td><code>Util::Array&lt; IndexT &gt; FindRowIndicesByAttr (const Attribute &amp;attr, bool firstMatchOnly) const</code></td>
<td>find all matching row indices by attribute value</td>
</tr>
<tr>
<td><code>Util::Array&lt; IndexT &gt; FindRowIndicesByAttrs (const Util::Array&lt; Attribute &gt; &amp;attrs, bool firstMatchOnly) const</code></td>
<td>find all matching row indices by multiple attribute values</td>
</tr>
<tr>
<td><code>IndexT FindRowIndexByAttr (const Attribute &amp;attr)</code></td>
<td>find all matching row indices by attribute value</td>
</tr>
<tr>
<td><code>IndexT FindRowIndexByAttrs (const Util::Array&lt; Attribute &gt; &amp;attrs)</code></td>
<td>find all matching row indices by multiple attribute values</td>
</tr>
<tr>
<td><code>void SetRowUserData (IndexT rowIndex, void *p)</code></td>
<td>set an optional row user data pointer</td>
</tr>
<tr>
<td><code>void * GetRowUserData (IndexT rowIndex)</code></td>
<td>get optional row user data pointer</td>
</tr>
<tr>
<td><code>void SetAttr (const Attr::Attribute &amp;attr, IndexT rowIndex)</code></td>
<td>set a generic attribute (slow!)</td>
</tr>
<tr>
<td><code>void SetBool (const BoolAttrId &amp;colAttrId, IndexT</code></td>
<td>set a boolean attribute (fast!)</td>
</tr>
<tr>
<td>Function</td>
<td>Parameters</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>bool GetBool</td>
<td>(const BoolAttrId &amp;colAttrId, IndexT rowIndex)</td>
</tr>
<tr>
<td>float GetFloat</td>
<td>(const FloatAttrId &amp;colAttrId, IndexT rowIndex)</td>
</tr>
<tr>
<td>void SetFloat</td>
<td>(const FloatAttrId &amp;colAttrId, IndexT rowIndex, float val)</td>
</tr>
<tr>
<td>int GetInt</td>
<td>(const IntAttrId &amp;colAttrId, IndexT rowIndex)</td>
</tr>
<tr>
<td>void SetInt</td>
<td>(const IntAttrId &amp;colAttrId, IndexT rowIndex, int val)</td>
</tr>
<tr>
<td>const Util::String &amp; GetString</td>
<td>(const StringAttrId &amp;colAttrId, IndexT rowIndex)</td>
</tr>
<tr>
<td>void SetString</td>
<td>(const StringAttrId &amp;colAttrId, IndexT rowIndex, const Util::String &amp;val)</td>
</tr>
<tr>
<td>Math::float4 GetFloat4</td>
<td>(const float4AttrId &amp;colAttrId, IndexT rowIndex)</td>
</tr>
<tr>
<td>void SetFloat4</td>
<td>(const float4AttrId &amp;colAttrId, IndexT rowIndex, const Math::float4 &amp;val)</td>
</tr>
<tr>
<td>Math::matrix44 GetMatrix44</td>
<td>(const Matrix44AttrId &amp;colAttrId, IndexT rowIndex)</td>
</tr>
<tr>
<td>void SetMatrix44</td>
<td>(const Matrix44AttrId &amp;colAttrId, IndexT rowIndex, const Math::matrix44 &amp;val)</td>
</tr>
<tr>
<td>void SetGuid</td>
<td>(const GuidAttrId &amp;colAttrId, IndexT rowIndex)</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>set guid value</td>
<td>const <code>Util::Guid</code> &amp;guid</td>
</tr>
<tr>
<td>get guid value</td>
<td></td>
</tr>
<tr>
<td>set blob value</td>
<td>const <code>Util::Blob</code> &amp;blob</td>
</tr>
<tr>
<td>get blob value</td>
<td></td>
</tr>
<tr>
<td>set bool value by column index</td>
<td>bool</td>
</tr>
<tr>
<td>get bool value by column index</td>
<td></td>
</tr>
<tr>
<td>set float value by column index</td>
<td>float</td>
</tr>
<tr>
<td>get float value by column index</td>
<td></td>
</tr>
<tr>
<td>set int value by column index</td>
<td>int</td>
</tr>
<tr>
<td>get int value by column index</td>
<td></td>
</tr>
<tr>
<td>set string value by column index</td>
<td>const <code>Util::String</code> &amp;val</td>
</tr>
<tr>
<td>get string value by column index</td>
<td></td>
</tr>
<tr>
<td>set float4 value by column index</td>
<td>const <code>Math::float4</code> &amp;val</td>
</tr>
<tr>
<td>get float4 value by column index</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>void</strong> SetMatrix44** (IndexT colIndex, IndexT rowIndex, const Math::matrix44 &amp;val)**</td>
<td>set matrix44 value by column index</td>
</tr>
<tr>
<td><strong>Math::matrix44</strong> GetMatrix44** (IndexT colIndex, IndexT rowIndex)** const**</td>
<td>get matrix44 value by column index</td>
</tr>
<tr>
<td><strong>void</strong> SetGuid** (IndexT colIndex, IndexT rowIndex, const Util::Guid &amp;guid)**</td>
<td>set guid value by column index</td>
</tr>
<tr>
<td><strong>const Util::Guid</strong> &amp; GetGuid** (IndexT colIndex, IndexT rowIndex)** const**</td>
<td>get guid value by column index</td>
</tr>
<tr>
<td><strong>void</strong> SetBlob** (IndexT colIndex, IndexT rowIndex, const Util::Blob &amp;blob)**</td>
<td>set blob value by column index</td>
</tr>
<tr>
<td><strong>const Util::Blob</strong> &amp; GetBlob** (IndexT colIndex, IndexT rowIndex)** const**</td>
<td>get blob value by column index</td>
</tr>
<tr>
<td><strong>bool</strong> LoadXmlTable** (const Util::String &amp;fileName)**</td>
<td>load xml table</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 Rtti &amp;rtti)** const**</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool</strong> 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><strong>bool</strong> 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><strong>bool</strong> 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><strong>bool</strong> 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><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>
</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>
</table>

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

void Attr::AttributeTable::ResetModifiedState()
reset all the modified bits in the table

This method resets the object to the unmodified state, which means the new-row and new-column index arrays are reset, and all modified bits are cleared.

void Attr::AttributeTable::Clear()
clear the table object

Clear all rows data and reset the number of rows.

void Attr::AttributeTable::PrintDebug()
print the contents of the table for debugging reasons

Print contents of the table for debugging.

void Attr::AttributeTable::BeginAddColumns(bool recordAsNewColumns = true)
optional: call before adding columns, speeds up adding many columns at once

Begin adding columns. Columns can be added at any time, but it will be much more efficient when called between BeginAddColumns() and EndAddColumns(), since this will save a lot of re-allocations.

void Attr::AttributeTable::AddColumn(const AttrId &id, & bool recordAsNewColumn = true)
add a column

Add a column to the attribute table. If the attribute table already contains data, this will reallocate the existing data buffer. The name, data type, access mode, etc of the column is all defined by the given attribute id. The new column will be filled with the attribute id's default value.

```cpp
void Attr::AttributeTable::EndAddColumns()
```

optional: call after adding columns, speeds up adding many columns at once

End adding columns. This will do the actual work.

```cpp
IndexT Attr::AttributeTable::AddRow()
```

add a row to the table, returns index of new row

Adds an empty row at the end of the value buffer. The row will be marked invalid until the first value is set in the row. This will re-allocate the existing value buffer. If you know beforehand how many rows will exist in the table it is more efficient to use one SetNumRows(N) instead of N times `AddRow()`! The method returns the index of the newly added row. The row will be filled with the row attribute's default values.

```cpp
void Attr::AttributeTable::ClearNewRowFlags()
```

clear the new row flags, so that new rows are treated like updated rows

Clears the new row flags. After this, new rows are treated just like updated rows, which may be useful for some database operations (when an UPDATE is wanted instead of an INSERT).

```cpp
void Attr::AttributeTable::ClearDeletedRowsFlags()
```
clear deleted rows flags

Clears the deleted row flags.

```cpp
void
Attr::AttributeTable::DeleteRow(IndexT rowIndex)
```

mark a row as deleted from the table

This marks a row for deletion. Note that the row will only be marked for deletion, deleted row indices are returned with the `GetDeletedRowIndices()` call. The row will never be physically removed from memory!

```cpp
void
Attr::AttributeTable::DeleteAllRows()
```

mark all rows as deleted

Marks all rows in the table as deleted.

```cpp
bool
Attr::AttributeTable::IsRowDeleted(IndexT rowIndex) const [inline]
```

return true if row has been marked as deleted

Return true if a row has been marked as deleted.

```cpp
IndexT
Attr::AttributeTable::CopyRow(IndexT fromRowIndex)
```

create a new row as copy of another row

This creates a new row as a copy of an existing row. Returns the index of the new row. NOTE: the user data will be initialized to 0 for the new row!

```cpp
IndexT
Attr::AttributeTable::CopyExtRow(* AttributeTable other,
    IndexT otherRowIndex,
    bool createMissingRows = false)
```
create a new row as copy of a row from another value table

Create a new row as a copy of a row in another value table. The layouts of the value tables must not match, since only matching columns will be considered. NOTE: the user data will be initialised to 0 for the new row.

```cpp
bool Attr::AttributeTable::IsRowModified(IndexT rowIndex) const [inline]

return true if a row has been modified since the last ResetModifiedState()
```

Quickly check if a row is modified.

```cpp
Util::Array<IndexT> Attr::AttributeTable::GetModifiedRowsExcludeNewAndDeletedRows() const

return array of modified rows, exclude rows marked as rows
```

Returns an array of all modified rows, but excludes rows marked as new.

```cpp
void Attr::AttributeTable::ReserveRows(SizeT num)

reserve rows to reduce re-allocation overhead
```

Reserve N more rows beforehand to reduce re-allocation overhead during AddRow().

```cpp
Util::Array<IndexT> Attr::AttributeTable::FindRowIndicesByAttr(const Attribute attr, & bool firstMatchOnly) const

find all matching row indices by attribute value
```

Finds multiple row indices by matching attribute. This method can be
slow since it may search linearly (and vertically) through the table.

```cpp
Util::Array< IndexT > Util::Array< attr >
Attr::AttributeTable::FindRowIndicesByAttrs ( const
&
bool firstMatchOnly )
```

find all matching row indices by multiple attribute values

Finds multiple row indices by multiple matching attribute. This method can be slow since it may search linearly (and vertically) through the table.

```cpp
const
IndexT
Attr::AttributeTable::FindRowIndexByAttr ( Attribute attr )
```

find all matching row indices by attribute value

Finds single row index by matching attribute. This method can be slow since it may search linearly (and vertically) through the table.

```cpp
const
IndexT
Attr::AttributeTable::FindRowIndexByAttrs ( Util::Array< attr >
&
)
```

find all matching row indices by multiple attribute values

Finds single row index by multiple matching attributes. This method can be slow since it may search linearly (and vertically) through the table.

```cpp
void
Attr::AttributeTable::SetAttr ( Attribute attr,
&
IndexT rowIndex )
```

set a generic attribute (slow!)
Set a generic attribute, this is a slow 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.

```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

Attr::AttrId
#include <attrid.h>

Inheritance diagram for Attr::AttrId:
Detailed Description

An attribute ID is used to carry attribute types (no values) around. **Attribute** IDs are compile-time safe, since each attribute ID represents its own C++ type.

(C) 2006 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AttrId ()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>AttrId (const AttrId &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>AttrId (const AttributeDefinitionBase &amp;def)</code></td>
<td>construct from attribute definition class</td>
</tr>
<tr>
<td><code>AttrId (const AttributeDefinitionBase *ptr)</code></td>
<td>construct from pointer to definition class</td>
</tr>
<tr>
<td><code>AttrId (const Util::String &amp;name)</code></td>
<td>construct from attribute name</td>
</tr>
<tr>
<td><code>AttrId (const Util::FourCC &amp;fcc)</code></td>
<td>construct from attribute fourcc code</td>
</tr>
<tr>
<td><code>bool operator==(const AttrId &amp;rhs) const</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const AttrId &amp;rhs) const</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>bool operator&gt;(const AttrId &amp;rhs) const</code></td>
<td>greater operator</td>
</tr>
<tr>
<td><code>bool operator&gt;=(const AttrId &amp;rhs) const</code></td>
<td>greater-or-equal operator</td>
</tr>
<tr>
<td><code>bool operator&lt;(const AttrId &amp;rhs) const</code></td>
<td>lesser operator</td>
</tr>
<tr>
<td><code>bool operator&lt;=(const AttrId &amp;rhs) const</code></td>
<td>lesser-or-equal operator</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetName () const</code></td>
<td>get attribute name</td>
</tr>
<tr>
<td><code>const Util::FourCC &amp; GetFourCC () const</code></td>
<td>get fourcc code</td>
</tr>
<tr>
<td><code>AccessMode GetAccessMode () const</code></td>
<td>get access type</td>
</tr>
<tr>
<td><code>ValueType GetValueType () const</code></td>
<td>get value type</td>
</tr>
<tr>
<td><code>bool IsDynamic () const</code></td>
<td></td>
</tr>
</tbody>
</table>
bool IsValid () const
return true if the attribute id object is valid

bool GetBoolDefValue () const
get bool default value

int GetIntDefValue () const
get int default value

float GetFloatDefValue () const
get float default value

const Util::String & GetStringDefValue () const
get string default value

Math::float4 GetFloat4DefValue () const
get float4 default value

const Math::matrix44 & GetMatrix44DefValue () const
get matrix44 default value

const Util::Blob & GetBlobDefValue () const
get blob default value

const Util::Guid & GetGuidDefValue () const
get guid default value
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isValidName</code> (const <code>Util::String</code> &amp;n)</td>
<td><code>return true if the provided attribute id name is valid</code></td>
</tr>
<tr>
<td><code>isValidFourCC</code> (const <code>Util::FourCC</code> &amp;fcc)</td>
<td><code>return true if the provided attribute id fourcc is valid</code></td>
</tr>
<tr>
<td><code>Util::FixedArray&lt;AttrId&gt; GetAllAttrIds()</code></td>
<td><code>return all attribute id's (slow!)</code></td>
</tr>
</tbody>
</table>
Member Function Documentation

Util::FixedArray<
AttrId > ( ) [static]
Attr::AttrId::GetAllAttrIds

return all attribute id's (slow!)

Returns an array with all existing attribute ids. This is a slow method.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Attr::BlobAttrId
Attr::BlobAttrId Class Reference

#include <blobattrid.h>

Inheritance diagram for Attr::BlobAttrId:
Detailed Description

Typed attribute id for blob type.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BlobAttrId</strong> (const <code>AttrId</code> &amp;rhs)</td>
<td>construct from generic attribute id</td>
</tr>
<tr>
<td><strong>BlobAttrId</strong> (const <code>AttributeDefinition&lt; BlobTypeClass, const Util::Blob &amp; &gt;</code> &amp;rhs)</td>
<td>construct from attribute definition</td>
</tr>
<tr>
<td><strong>BlobAttrId</strong> (const <code>Util::String</code> &amp;rhs)</td>
<td>construct from name</td>
</tr>
<tr>
<td><strong>BlobAttrId</strong> (const <code>Util::FourCC</code> &amp;rhs)</td>
<td>construct from fourcc code</td>
</tr>
<tr>
<td>bool <code>operator==</code> (const <code>BlobAttrId</code> &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool <code>operator!=</code> (const <code>BlobAttrId</code> &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td>bool <code>operator==</code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool <code>operator!=</code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td>bool <code>operator&gt; </code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>greater operator</td>
</tr>
<tr>
<td>bool <code>operator&gt;&gt;=</code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>greater-or-equal operator</td>
</tr>
<tr>
<td>bool <code>operator&lt;</code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>lesser operator</td>
</tr>
<tr>
<td>bool <code>operator&lt;=</code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>lesser-or-equal operator</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp; <code>GetName</code> () const</td>
<td>get attribute name</td>
</tr>
<tr>
<td>const <code>Util::FourCC</code> &amp; <code>GetFourCC</code> () const</td>
<td>get fourcc code</td>
</tr>
<tr>
<td><code>AccessMode</code> <code>GetAccessMode</code> () const</td>
<td>get access type</td>
</tr>
<tr>
<td><code>ValueType</code> <code>GetValueType</code> () const</td>
<td>get value type</td>
</tr>
<tr>
<td>Type</td>
<td>Method</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsDynamic()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsValid()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>GetBoolDefValue()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>GetIntDefValue()</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>GetFloatDefValue()</code> const</td>
</tr>
<tr>
<td>const</td>
<td><code>Util::String &amp; GetStringDefValue()</code> const</td>
</tr>
<tr>
<td>const</td>
<td><code>Math::float4 GetFloat4DefValue()</code> const</td>
</tr>
<tr>
<td>const</td>
<td><code>Math::matrix44 GetMatrix44DefValue()</code> const</td>
</tr>
<tr>
<td>const</td>
<td><code>Util::Blob GetBlobDefValue()</code> const</td>
</tr>
<tr>
<td>const</td>
<td><code>Util::Guid GetGuidDefValue()</code> const</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsValidName (const Util::String &amp;n)</code></td>
<td>return true if the provided attribute id name is valid</td>
</tr>
<tr>
<td><code>IsValidFourCC (const Util::FourCC &amp;fcc)</code></td>
<td>return true if the provided attribute id fourcc is valid</td>
</tr>
<tr>
<td><code>Util::FixedArray &lt; AttrId &gt; GetAllAttrIds ()</code></td>
<td>return all attribute id's (slow!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

Util::FixedArray<
  AttrId >
Attr::AttrId::GetAllAttrIds

return all attribute id's (slow!)

Returns an array with all existing attribute ids. This is a slow method.
**Attr::BoolAttrId**
Attr::BoolAttrId Class Reference

#include <boolattrid.h>

Inheritance diagram for Attr::BoolAttrId:
Detailed Description

Typed attribute id for bool type.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BoolAttrId</strong> (const <code>AttrId</code> &amp;rhs)</td>
<td>construct from generic attribute id</td>
</tr>
<tr>
<td><strong>BoolAttrId</strong> (const <code>AttributeDefinition&lt;BoolTypeClass, bool&gt;</code> &amp;rhs)</td>
<td>construct from attribute definition</td>
</tr>
<tr>
<td><strong>BoolAttrId</strong> (const <code>Util::String</code> &amp;rhs)</td>
<td>construct from name</td>
</tr>
<tr>
<td><strong>BoolAttrId</strong> (const <code>Util::FourCC</code> &amp;rhs)</td>
<td>construct from fourcc code</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>operator==</code> (const <code>BoolAttrId</code> &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td><code>operator!=</code> (const <code>BoolAttrId</code> &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>operator==</code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td><code>operator!=</code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>operator&gt;</code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>greater operator</td>
</tr>
<tr>
<td><code>operator&gt;&gt;=</code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>greater-or-equal operator</td>
</tr>
<tr>
<td><code>operator&lt;</code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>lesser operator</td>
</tr>
<tr>
<td><code>operator&lt;=</code> (const <code>AttrId</code> &amp;rhs) const</td>
<td>lesser-or-equal operator</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetFourCC()</code> (const <code>Util::FourCC</code> &amp;</td>
<td>get fourcc code</td>
</tr>
<tr>
<td><code>GetName()</code> () const</td>
<td>get attribute name</td>
</tr>
<tr>
<td><code>GetAccessMode()</code> () const</td>
<td>get access type</td>
</tr>
<tr>
<td><code>GetValueType()</code> () const</td>
<td>get value type</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><strong>IsDynamic()</strong> const</td>
<td>return true if this attribute was dynamically registered</td>
</tr>
<tr>
<td><strong>IsValid()</strong> const</td>
<td>return true if the attribute id object is valid</td>
</tr>
<tr>
<td><strong>GetBoolDefValue()</strong> const</td>
<td>get bool default value</td>
</tr>
<tr>
<td><strong>GetIntDefValue()</strong> const</td>
<td>get int default value</td>
</tr>
<tr>
<td><strong>GetFloatDefValue()</strong> const</td>
<td>get float default value</td>
</tr>
<tr>
<td><strong>GetStringDefValue()</strong> const</td>
<td>get string default value</td>
</tr>
<tr>
<td><strong>GetFloat4DefValue()</strong> const</td>
<td>get float4 default value</td>
</tr>
<tr>
<td><strong>GetMatrix44DefValue()</strong> const</td>
<td>get matrix44 default value</td>
</tr>
<tr>
<td><strong>GetBlobDefValue()</strong> const</td>
<td>get blob default value</td>
</tr>
<tr>
<td><strong>GetGuidDefValue()</strong> const</td>
<td>get guid default value</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsValidName</code> (const <code>Util::String</code> &amp;n)</td>
<td>return true if the provided attribute id name is valid</td>
</tr>
<tr>
<td><code>IsValidFourCC</code> (const <code>Util::FourCC</code> &amp;fcc)</td>
<td>return true if the provided attribute id fourcc is valid</td>
</tr>
<tr>
<td><code>getAllAttrIds</code> ()</td>
<td>return all attribute id's (slow!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

Util::FixedArray<
AttrId > ( ) [static, inherited]
Attr::AttrId:: GetAllAttrIds

return all attribute id's (slow!)

Returns an array with all existing attribute ids. This is a slow method.
Attr::Float4AttrId
#Attr::Float4AttrId Class Reference

#include <float4attrid.h>

Inheritance diagram for Attr::Float4AttrId:

```
Class

Attr::AttrId

attr::Float4AttrId
```
Detailed Description

Typed attribute id for float4 type.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Float4AttrId</strong> (const AttrId &amp;rhs)</td>
<td>construct from generic attribute id</td>
</tr>
<tr>
<td><strong>Float4AttrId</strong> (const AttributeDefinition&lt; Float4TypeClass, const Math::float4 &amp; &gt; &amp;rhs)</td>
<td>construct from attribute definition</td>
</tr>
<tr>
<td><strong>Float4AttrId</strong> (const Util::String &amp;rhs)</td>
<td>construct from name</td>
</tr>
<tr>
<td><strong>Float4AttrId</strong> (const Util::FourCC &amp;rhs)</td>
<td>construct from fourcc code</td>
</tr>
<tr>
<td>bool operator== (const Float4AttrId &amp;rhs)</td>
<td>const equality operator</td>
</tr>
<tr>
<td>bool operator!= (const Float4AttrId &amp;rhs)</td>
<td>const inequality operator</td>
</tr>
<tr>
<td>bool operator==(const AttrId &amp;rhs)</td>
<td>const equality operator</td>
</tr>
<tr>
<td>bool operator!=(const AttrId &amp;rhs)</td>
<td>const inequality operator</td>
</tr>
<tr>
<td>bool operator&gt; (const AttrId &amp;rhs)</td>
<td>const greater operator</td>
</tr>
<tr>
<td>bool operator&gt;= (const AttrId &amp;rhs)</td>
<td>const greater-or-equal operator</td>
</tr>
<tr>
<td>bool operator&lt; (const AttrId &amp;rhs)</td>
<td>const lesser operator</td>
</tr>
<tr>
<td>bool operator&lt;= (const AttrId &amp;rhs)</td>
<td>const lesser-or-equal operator</td>
</tr>
<tr>
<td>const Util::String &amp; GetName ()</td>
<td>const get attribute name</td>
</tr>
<tr>
<td>const Util::FourCC &amp; GetFourCC ()</td>
<td>const get fourcc code</td>
</tr>
<tr>
<td>AccessMode &amp; GetAccessMode ()</td>
<td>const get access type</td>
</tr>
<tr>
<td>ValueType &amp; GetValueType ()</td>
<td>const get value type</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsDynamic</strong> () const</td>
</tr>
<tr>
<td>------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td>return true if this attribute was dynamically registered</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsValid</strong> () const</td>
</tr>
<tr>
<td></td>
<td>return true if the attribute id object is valid</td>
</tr>
<tr>
<td>bool</td>
<td><strong>GetBoolDefValue</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get bool default value</td>
</tr>
<tr>
<td>int</td>
<td><strong>GetIntDefValue</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get int default value</td>
</tr>
<tr>
<td>float</td>
<td><strong>GetFloatDefValue</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get float default value</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><strong>GetStringDefValue</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get string default value</td>
</tr>
<tr>
<td>const Math::float4 &amp;</td>
<td><strong>GetFloat4DefValue</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get float4 default value</td>
</tr>
<tr>
<td>const Math::matrix44 &amp;</td>
<td><strong>GetMatrix44DefValue</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get matrix44 default value</td>
</tr>
<tr>
<td>const Util::Blob &amp;</td>
<td><strong>GetBlobDefValue</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get blob default value</td>
</tr>
<tr>
<td>const Util::Guid &amp;</td>
<td><strong>GetGuidDefValue</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get guid default value</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isValidName</code> (const <code>Util::String &amp;n</code>)</td>
<td><code>return true if the provided attribute id name is valid</code></td>
</tr>
<tr>
<td><code>isValidFourCC</code> (const <code>Util::FourCC &amp;fcc</code>)</td>
<td><code>return true if the provided attribute id fourcc is valid</code></td>
</tr>
<tr>
<td><code>getAllAttrIds()</code></td>
<td><code>return all attribute id's (slow!)</code></td>
</tr>
</tbody>
</table>
Member Function Documentation

Util::FixedArray<
AttrId > ( ) [static, inherited]
Attr::AttrId::GetAllAttrIds

return all attribute id's (slow!)

Returns an array with all existing attribute ids. This is a slow method.
Attr::FloatAttrId
Attr::FloatAttrId Class Reference

#include <floatattrid.h>

Inheritance diagram for Attr::FloatAttrId:

```
Attr::AttrId

Attr::FloatAttrId
```

Detailed Description

Typed attribute id for float type.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FloatAttrId (const AttrId &amp;rhs)</code></td>
<td>Construct from generic attribute id</td>
</tr>
<tr>
<td><code>FloatAttrId (const AttributeDefinition&lt;FloatTypeClass, float&gt; &amp;rhs)</code></td>
<td>Construct from attribute definition</td>
</tr>
<tr>
<td><code>FloatAttrId (const Util::String &amp;rhs)</code></td>
<td>Construct from name</td>
</tr>
<tr>
<td><code>FloatAttrId (const Util::FourCC &amp;rhs)</code></td>
<td>Construct from fourcc code</td>
</tr>
<tr>
<td>bool <code>operator== (const FloatAttrId &amp;rhs)</code> const</td>
<td>Equality operator</td>
</tr>
<tr>
<td>bool <code>operator!=(const FloatAttrId &amp;rhs)</code> const</td>
<td>Inequality operator</td>
</tr>
<tr>
<td>bool <code>operator==(const AttrId &amp;rhs)</code> const</td>
<td>Equality operator</td>
</tr>
<tr>
<td>bool <code>operator!=(const AttrId &amp;rhs)</code> const</td>
<td>Inequality operator</td>
</tr>
<tr>
<td>bool <code>operator&gt; (const AttrId &amp;rhs)</code> const</td>
<td>Greater operator</td>
</tr>
<tr>
<td>bool <code>operator&gt;= (const AttrId &amp;rhs)</code> const</td>
<td>Greater-or-equal operator</td>
</tr>
<tr>
<td>bool <code>operator&lt; (const AttrId &amp;rhs)</code> const</td>
<td>Lesser operator</td>
</tr>
<tr>
<td>bool <code>operator&lt;= (const AttrId &amp;rhs)</code> const</td>
<td>Lesser-or-equal operator</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetName () const</code></td>
<td>Get attribute name</td>
</tr>
<tr>
<td><code>const Util::FourCC &amp; GetFourCC () const</code></td>
<td>Get fourcc code</td>
</tr>
<tr>
<td><code>AccessMode GetAccessMode () const</code></td>
<td>Get access type</td>
</tr>
<tr>
<td><code>ValueType GetValueType () const</code></td>
<td>Get value type</td>
</tr>
<tr>
<td>return type</td>
<td>method name</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>bool</td>
<td>IsDynamic () const</td>
</tr>
<tr>
<td>bool</td>
<td>IsValid () const</td>
</tr>
<tr>
<td>bool</td>
<td>GetBoolDefValue () const</td>
</tr>
<tr>
<td>int</td>
<td>GetIntDefValue () const</td>
</tr>
<tr>
<td>float</td>
<td>GetFloatDefValue () const</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetStringDefValue () const</td>
</tr>
<tr>
<td>Math::float4</td>
<td>GetFloat4DefValue () const</td>
</tr>
<tr>
<td>const Math::matrix44 &amp;</td>
<td>GetMatrix44DefValue () const</td>
</tr>
<tr>
<td>const Util::Blob &amp;</td>
<td>GetBlobDefValue () const</td>
</tr>
<tr>
<td>const Util::Guid &amp;</td>
<td>GetGuidDefValue () const</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isValidName</code> (const <code>Util::String</code> &amp;n)</td>
<td>return true if the provided attribute id name is valid</td>
</tr>
<tr>
<td><code>isValidFourCC</code> (const <code>Util::FourCC</code> &amp;fcc)</td>
<td>return true if the provided attribute id fourcc is valid</td>
</tr>
<tr>
<td><code>GetAllAttrIds</code> ()</td>
<td>return all attribute id's (slow!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Util::FixedArray< AttrId >**
Attr::AttrId::GetAllAttrIds

return all attribute id's (slow!)

Returns an array with all existing attribute ids. This is a slow method.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

Attr::GuidAttrId
#include <guidattrid.h>

Inheritance diagram for Attr::GuidAttrId:

```
```

---
Detailed Description

Typed attribute id for guid type.

(C) 2006 Radon Labs GmbH
Public Member Functions

**GuidAttrId** (const AttrId &rhs)
construct from generic attribute id

**GuidAttrId** (const AttributeDefinition<
GuidTypeClass, const Util::Guid &> &rhs)
construct from attribute definition

**GuidAttrId** (const Util::String &rhs)
construct from name

**GuidAttrId** (const Util::FourCC &rhs)
construct from fourcc code

bool operator==(const GuidAttrId &rhs) const
equality operator

bool operator!=(const GuidAttrId &rhs) const
inequality operator

bool operator==(const AttrId &rhs) const
equality operator

bool operator!=(const AttrId &rhs) const
inequality operator

bool operator>(const AttrId &rhs) const
greater operator

bool operator>=(const AttrId &rhs) const
greater-or-equal operator

bool operator<(const AttrId &rhs) const
lesser operator

bool operator<=(const AttrId &rhs) const
lesser-or-equal operator

const Util::String & GetName () const
get attribute name

const Util::FourCC & GetFourCC () const
get fourcc code

AccessMode GetAccessMode () const
get access type

ValueType GetValueType () const
get value type
<table>
<thead>
<tr>
<th>bool</th>
<th>IsDynamic () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if this attribute was dynamically registered</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>IsValid () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if the attribute id object is valid</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th>GetBoolDefValue () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get bool default value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>GetIntDefValue () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get int default value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>float</th>
<th>GetFloatDefValue () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get float default value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::String &amp;</th>
<th>GetStringDefValue () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get string default value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Math::float4</th>
<th>GetFloat4DefValue () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get float4 default value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Math::matrix44 &amp;</th>
<th>GetMatrix44DefValue () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get matrix44 default value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::Blob &amp;</th>
<th>GetBlobDefValue () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get blob default value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>const Util::Guid &amp;</th>
<th>GetGuidDefValue () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get guid default value</td>
</tr>
</tbody>
</table>
# Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isValidName</code> (const Util::String &amp;n)</td>
<td>return true if the provided attribute id name is valid</td>
</tr>
<tr>
<td><code>isValidFourCC</code> (const Util::FourCC &amp;fcc)</td>
<td>return true if the provided attribute id fourcc is valid</td>
</tr>
<tr>
<td><code>getAllAttrIds</code></td>
<td>return all attribute id's (slow!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Util::FixedArray< AttrId > ( ) [static, inherited]**
Attr::AttrId::GetAllAttrIds

return all attribute id's (slow!)

Returns an array with all existing attribute ids. This is a slow method.
Attr::IntAttrId
Attr::IntAttrId Class Reference

#include <intattrid.h>

Inheritance diagram for Attr::IntAttrId:

```
Attr::AttrId

Attr::IntAttrId
```
Detailed Description

Typed attribute id for integer type.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IntAttrId (const AttrId &amp;rhs)</code></td>
<td>construct from generic attribute id</td>
</tr>
</tbody>
</table>
| `IntAttrId (const AttributeDefinition<
IntTypeClass, int > &rhs)` | construct from attribute definition                   |
<p>| <code>IntAttrId (const Util::String &amp;rhs)</code> | construct from name                                   |
| <code>IntAttrId (const Util::FourCC &amp;rhs)</code> | construct from fourcc code                            |
| <code>bool operator== (const IntAttrId &amp;rhs) const</code> | equality operator                                     |
| <code>bool operator!=(const IntAttrId &amp;rhs) const</code> | equality operator                                     |
| <code>bool operator==(const AttrId &amp;rhs) const</code> | equality operator                                     |
| <code>bool operator!=(const AttrId &amp;rhs) const</code> | equality operator                                     |
| <code>bool operator&gt; (const AttrId &amp;rhs) const</code> | greater operator                                      |
| <code>bool operator&gt;=(const AttrId &amp;rhs) const</code> | greater-or-equal operator                             |
| <code>bool operator&lt; (const AttrId &amp;rhs) const</code> | lesser operator                                       |
| <code>bool operator&lt;= (const AttrId &amp;rhs) const</code> | lesser-or-equal operator                              |
| <code>const Util::String &amp; GetName () const</code> | get attribute name                                    |
| <code>const Util::FourCC &amp; GetFourCC () const</code> | get fourcc code                                       |
| <code>AccessMode GetAccessMode () const</code> | get access type                                       |
| <code>ValueType GetValueType () const</code> | get value type                                        |</p>
<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>IsDynamic () const</td>
<td>return true if this attribute was dynamically registered</td>
</tr>
<tr>
<td>bool</td>
<td>IsValid () const</td>
<td>return true if the attribute id object is valid</td>
</tr>
<tr>
<td>bool</td>
<td>GetBoolDefValue () const</td>
<td>get bool default value</td>
</tr>
<tr>
<td>int</td>
<td>GetIntDefValue () const</td>
<td>get int default value</td>
</tr>
<tr>
<td>float</td>
<td>GetFloatDefValue () const</td>
<td>get float default value</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetStringDefValue () const</td>
<td>get string default value</td>
</tr>
<tr>
<td>const Math::float4</td>
<td>GetFloat4DefValue () const</td>
<td>get float4 default value</td>
</tr>
<tr>
<td>const Math::matrix44 &amp;</td>
<td>GetMatrix44DefValue () const</td>
<td>get matrix44 default value</td>
</tr>
<tr>
<td>const Util::Blob &amp;</td>
<td>GetBlobDefValue () const</td>
<td>get blob default value</td>
</tr>
<tr>
<td>const Util::Guid &amp;</td>
<td>GetGuidDefValue () const</td>
<td>get guid default value</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isValidName (const Util::String &amp;n)</code></td>
<td>return true if the provided attribute id name is valid</td>
</tr>
<tr>
<td><code>isValidFourCC (const Util::FourCC &amp;fcc)</code></td>
<td>return true if the provided attribute id fourcc is valid</td>
</tr>
<tr>
<td><code>Util::FixedArray&lt;AttrId&gt; GetAllAttrIds ()</code></td>
<td>return all attribute id's (slow!)</td>
</tr>
</tbody>
</table>
Util::FixedArray<
AttrId > ( ) [static, inherited]
Attr::AttrId::GetAllAttrIds

return all attribute id's (slow!)

Returns an array with all existing attribute ids. This is a slow method.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Attr::Matrix44AttrId
Attr::Matrix44AttrId Class Reference

#include <matrix44attrid.h>

Inheritance diagram for Attr::Matrix44AttrId:
Detailed Description

Typed attribute id for matrix44 type.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Matrix44AttrId</strong> (const AttrId &amp;rhs)</td>
<td>construct from generic attribute id</td>
</tr>
<tr>
<td><strong>Matrix44AttrId</strong> (const AttributeDefinition&lt;Matrix44TypeClass, const Math::matrix44 &amp; &gt; &amp;rhs)</td>
<td>construct from attribute definition</td>
</tr>
<tr>
<td><strong>Matrix44AttrId</strong> (const Util::String &amp;rhs)</td>
<td>construct from name</td>
</tr>
<tr>
<td><strong>Matrix44AttrId</strong> (const Util::FourCC &amp;rhs)</td>
<td>construct from fourcc code</td>
</tr>
<tr>
<td>bool operator==(const Matrix44AttrId &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool operator!=(const Matrix44AttrId &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td>bool operator==(const AttrId &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool operator!=(const AttrId &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td>bool operator&gt;(const AttrId &amp;rhs) const</td>
<td>greater operator</td>
</tr>
<tr>
<td>bool operator&gt;=(const AttrId &amp;rhs) const</td>
<td>greater-or-equal operator</td>
</tr>
<tr>
<td>bool operator&lt;(const AttrId &amp;rhs) const</td>
<td>lesser operator</td>
</tr>
<tr>
<td>bool operator&lt;=(const AttrId &amp;rhs) const</td>
<td>lesser-or-equal operator</td>
</tr>
<tr>
<td>const Util::String &amp; GetName () const</td>
<td>get attribute name</td>
</tr>
<tr>
<td>const Util::FourCC &amp; GetFourCC () const</td>
<td>get fourcc code</td>
</tr>
<tr>
<td>AccessMode GetAccessMode () const</td>
<td>get access type</td>
</tr>
<tr>
<td>ValueType GetValueType () const</td>
<td></td>
</tr>
</tbody>
</table>
get value type

bool IsDynamic () const
return true if this attribute was dynamically registered

bool IsValid () const
return true if the attribute id object is valid

bool GetBoolDefValue () const
get bool default value

int GetIntDefValue () const
get int default value

float GetFloatDefValue () const
get float default value

const Util::String & GetStringDefValue () const
get string default value

Math::float4 GetFloat4DefValue () const
get float4 default value

const Math::matrix44 & GetMatrix44DefValue () const
get matrix44 default value

const Util::Blob & GetBlobDefValue () const
get blob default value

const Util::Guid & GetGuidDefValue () const
get guid default value
Static Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsValidName</td>
<td>static bool IsValidName (const Util::String &amp;n) return true if the provided attribute id name is valid</td>
</tr>
<tr>
<td>IsValidFourCC</td>
<td>static bool IsValidFourCC (const Util::FourCC &amp;fcc) return true if the provided attribute id fourcc is valid</td>
</tr>
<tr>
<td>GetAllAttrIds</td>
<td>static Util::FixedArray&lt;AttrId&gt; GetAllAttrIds () return all attribute id's (slow!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```
Util::FixedArray<
AttrId > ( ) [static, inherited]
Attr::AttrId::GetAllAttrIds
```

return all attribute id's (slow!)

Returns an array with all existing attribute ids. This is a slow method.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

Attr::StringAttrId
Attr::StringAttrId Class Reference

#include <stringattrid.h>

Inheritance diagram for Attr::StringAttrId:
Detailed Description

Typed attribute id for string type.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StringAttrId (const AttrId &amp;rhs)</code></td>
<td>construct from generic attribute id</td>
</tr>
<tr>
<td><code>StringAttrId (const AttributeDefinition&lt;StringTypeClass, const Util::String &amp; &gt; &amp;rhs)</code></td>
<td>construct from attribute definition</td>
</tr>
<tr>
<td><code>StringAttrId (const Util::String &amp;rhs)</code></td>
<td>construct from name</td>
</tr>
<tr>
<td><code>StringAttrId (const Util::FourCC &amp;rhs)</code></td>
<td>construct from fourcc code</td>
</tr>
<tr>
<td><code>bool operator==(const StringAttrId &amp;rhs)</code></td>
<td>const equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const StringAttrId &amp;rhs)</code></td>
<td>const inequality operator</td>
</tr>
<tr>
<td><code>bool operator==(const AttrId &amp;rhs)</code></td>
<td>const equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const AttrId &amp;rhs)</code></td>
<td>const inequality operator</td>
</tr>
<tr>
<td><code>bool operator&gt;(const AttrId &amp;rhs)</code></td>
<td>const greater operator</td>
</tr>
<tr>
<td><code>bool operator&gt;=(const AttrId &amp;rhs)</code></td>
<td>const greater-or-equal operator</td>
</tr>
<tr>
<td><code>bool operator&lt; (const AttrId &amp;rhs)</code></td>
<td>const lesser operator</td>
</tr>
<tr>
<td><code>bool operator&lt;=(const AttrId &amp;rhs)</code></td>
<td>const lesser-or-equal operator</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetName ()</code></td>
<td>const get attribute name</td>
</tr>
<tr>
<td><code>const Util::FourCC &amp; GetFourCC ()</code></td>
<td>const get fourcc code</td>
</tr>
<tr>
<td><code>AccessMode GetAccessMode ()</code></td>
<td>const get access type</td>
</tr>
<tr>
<td><code>ValueType GetValueType ()</code></td>
<td>const get value type</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsDynamic()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsValid()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>GetBoolDefValue()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>GetIntDefValue()</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>GetFloatDefValue()</code> const</td>
</tr>
<tr>
<td>const <code>Util::String</code> &amp;</td>
<td><code>GetStringDefValue()</code> const</td>
</tr>
<tr>
<td><code>Math::float4</code></td>
<td><code>GetFloat4DefValue()</code> const</td>
</tr>
<tr>
<td>const <code>Math::matrix44</code> &amp;</td>
<td><code>GetMatrix44DefValue()</code> const</td>
</tr>
<tr>
<td>const <code>Util::Blob</code> &amp;</td>
<td><code>GetBlobDefValue()</code> const</td>
</tr>
<tr>
<td>const <code>Util::Guid</code> &amp;</td>
<td><code>GetGuidDefValue()</code> const</td>
</tr>
</tbody>
</table>
# Static Public Member Functions

<table>
<thead>
<tr>
<th>static bool</th>
<th>IsValidName (const Util::String &amp;n)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if the provided attribute id name is valid</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static bool</th>
<th>IsValidFourCC (const Util::FourCC &amp;fcc)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if the provided attribute id fourcc is valid</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static</th>
<th>Util::FixedArray&lt;AttrId&gt;</th>
<th>GetAllAttrIds ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return all attribute id's (slow!)</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

Util::FixedArray<
AttrId > ( ) [static, inherited]
Attr::AttrId::GetAllAttrIds

return all attribute id's (slow!)

Returns an array with all existing attribute ids. This is a slow method.
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

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><strong>Month</strong></td>
</tr>
<tr>
<td></td>
<td><em>months enum</em></td>
</tr>
<tr>
<td>enum</td>
<td><strong>Weekday</strong></td>
</tr>
<tr>
<td></td>
<td><em>weekdays enum</em></td>
</tr>
<tr>
<td>typedef unsigned int</td>
<td><strong>Year</strong></td>
</tr>
<tr>
<td></td>
<td><em>typedefs</em></td>
</tr>
</tbody>
</table>
### Public Member Functions

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Timing::CalendarTime GetSystemTime()</code></td>
<td>get the current system time</td>
</tr>
<tr>
<td><code>Timing::CalendarTime GetLocalTime()</code></td>
<td>get the current local time</td>
</tr>
<tr>
<td><code>SystemTimeToFileTime</code></td>
<td>convert system time to file time</td>
</tr>
<tr>
<td><code>FileTimeToSystemTime</code></td>
<td>convert file time to system time</td>
</tr>
<tr>
<td><code>LocalTimeToFileTime</code></td>
<td>convert local time to file time</td>
</tr>
<tr>
<td><code>FileTimeToLocalTime</code></td>
<td>convert file time to local time</td>
</tr>
<tr>
<td><code>Format</code></td>
<td>format to string</td>
</tr>
<tr>
<td><code>MonthToString</code></td>
<td>convert month to string</td>
</tr>
<tr>
<td><code>StringToMonth</code></td>
<td>convert string to month</td>
</tr>
<tr>
<td><code>WeekdayToString</code></td>
<td>convert weekday to string</td>
</tr>
<tr>
<td><code>StringToWeekday</code></td>
<td>convert string to weekday</td>
</tr>
</tbody>
</table>
Member Function Documentation

String Base::CalendarTimeBase::Format (const Util::String & fmtString,
const Timing::CalendarTime calTime &)

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

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:42 2008
Base::D3D9TransformDevice
Base::D3D9TransformDevice Class Reference

#include <d3d9transformdevice.h>
Detailed Description

(C) 2007 Radon Labs GmbH

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:42 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

**Base::DisplayDeviceBase**
#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><code>DisplayDeviceBase ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>virtual ~DisplayDeviceBase ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>bool <code>AdapterExists (CoreGraphics::Adapter::Code adapter)</code></td>
<td>return true if adapter exists</td>
</tr>
<tr>
<td>bool <code>GetAvailableDisplayModes</code> <a href="">CoreGraphics::DisplayMode</a> (CoreGraphics::Adapter::Code adapter, CoreGraphics::PixelFormat::Code pixelFormat)`</td>
<td>get available display modes on given adapter</td>
</tr>
<tr>
<td>bool <code>SupportsDisplayMode</code> (CoreGraphics::Adapter::Code adapter, const CoreGraphics::DisplayMode &amp;requestedMode)`</td>
<td>return true if a given display mode is supported</td>
</tr>
<tr>
<td>CoreGraphics::DisplayMode <code>GetCurrentAdapterDisplayMode</code> (CoreGraphics::Adapter::Code adapter)`</td>
<td>get current adapter display mode (i.e. the desktop display mode)</td>
</tr>
<tr>
<td>CoreGraphics::AdapterInfo <code>GetAdapterInfo</code> (CoreGraphics::Adapter::Code adapter)`</td>
<td>get general info about display adapter</td>
</tr>
<tr>
<td>void <code>SetAdapter</code> (CoreGraphics::Adapter::Code a)`</td>
<td>set display adapter (make sure adapter exists!)</td>
</tr>
<tr>
<td>CoreGraphics::Adapter::Code <code>GetAdapter</code> () const</td>
<td>get display adapter</td>
</tr>
<tr>
<td>void <code>SetDisplayMode</code> (const CoreGraphics::DisplayMode &amp;m)`</td>
<td>set display mode</td>
</tr>
<tr>
<td>Function</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 (make sure the display mode is supported!)</td>
</tr>
<tr>
<td><code>void SetAntiAliasQuality (CoreGraphics::AntiAliasQuality::Code aa)</code></td>
<td>set antialias quality</td>
</tr>
<tr>
<td><code>const CoreGraphics::AntiAliasQuality::Code GetAntiAliasQuality () const</code></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 () 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 SetWindowTitle (const Util::String &amp;t)</code></td>
<td>set optional window title</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
const Util::String & GetWindowTitle () const
get window title string

bool Open ()
open the display

void Close ()
close the display

bool IsOpen () const
return true if display is currently open

void ProcessWindowMessages ()
process window system messages, call this method once per frame

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 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
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><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>bool</th>
<th>NotifyEventHandlers (const CoreGraphics::DisplayEvent &amp;e)</th>
</tr>
</thead>
</table>

*notify event handlers about an event*
Member Function Documentation

```cpp
bool Base::DisplayDeviceBase::AdapterExists ( CoreGraphics::Adapter::Code adapter )

return true if adapter exists

Checks if the given adapter exists.

Reimplemented in Direct3D9::D3D9DisplayDevice.
```

```cpp
Util::Array< DisplayMode >
Base::DisplayDeviceBase::GetAvailableDisplayModes ( CoreGraphics::Adapter::Code ada,
                                                       CoreGraphics::PixelFormat::Code pixe
                                                     )

get available display modes on given adapter

Returns the display modes on the given adapter in the given pixel format.

Reimplemented in Direct3D9::D3D9DisplayDevice.
```

```cpp
bool Base::DisplayDeviceBase::SupportsDisplayMode ( CoreGraphics::Adapter::Code adapter,
                                                     const CoreGraphics::DisplayMode requestedMo &
                                                   )

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.
```

```cpp
DisplayMode Base::DisplayDeviceBase::GetCurrentAdapterDisplayMode ( CoreGraphics::Adapter::Code ada,
                                                                      CoreGraphics::PixelFormat::Code pixe
                                                        )
```

```cpp
Util::Array< DisplayMode >
Base::DisplayDeviceBase::GetAvailableDisplayModes ( CoreGraphics::Adapter::Code ada,
                                                     CoreGraphics::PixelFormat::Code pixe
                                                     )
```

get available display modes on given adapter

Returns the display modes on the given adapter in the given pixel format.

Reimplemented in Direct3D9::D3D9DisplayDevice.
```

```cpp
bool Base::DisplayDeviceBase::SupportsDisplayMode ( CoreGraphics::Adapter::Code adapter,
                                                     const CoreGraphics::DisplayMode requestedMo &
                                                   )

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.
```

```cpp
DisplayMode Base::DisplayDeviceBase::GetCurrentAdapterDisplayMode ( CoreGraphics::Adapter::Code ada,
                                                                      CoreGraphics::PixelFormat::Code pixe
                                                        )
```

```cpp
Util::Array< DisplayMode >
Base::DisplayDeviceBase::GetAvailableDisplayModes ( CoreGraphics::Adapter::Code ada,
                                                     CoreGraphics::PixelFormat::Code pixe
                                                     )
```

get available display modes on given adapter

Returns the display modes on the given adapter in the given pixel format.

Reimplemented in Direct3D9::D3D9DisplayDevice.
```

```cpp
bool Base::DisplayDeviceBase::SupportsDisplayMode ( CoreGraphics::Adapter::Code adapter,
                                                     const CoreGraphics::DisplayMode requestedMo &
                                                   )

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.
```

```cpp
DisplayMode Base::DisplayDeviceBase::GetCurrentAdapterDisplayMode ( CoreGraphics::Adapter::Code ada,
                                                                      CoreGraphics::PixelFormat::Code pixe
                                                        )
```
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**.

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

get general info about display adapter

Returns information about the provided adapter.

Reimplemented in **Direct3D9::D3D9DisplayDevice**.

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

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.

```c
bool Base::DisplayDeviceBase::Open ( )
```

open the display

Open the display.

Reimplemented in **Win32::Win32DisplayDevice**.

```c
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::GamePadBase
Base::GamePadBase Class Reference

#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><strong>Button</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>gamepad buttons</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>virtual ~GamePadBase ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>bool IsConnected () const</td>
<td>return true if this game pad is currently connected</td>
</tr>
<tr>
<td>IndexT GetPlayerIndex () const</td>
<td>get the player index of this game pad</td>
</tr>
<tr>
<td>bool ButtonPressed (Button btn) const</td>
<td>return true if a button is currently pressed</td>
</tr>
<tr>
<td>bool ButtonDown (Button btn) const</td>
<td>return true if button was down at least once in current frame</td>
</tr>
<tr>
<td>bool ButtonUp (Button btn) const</td>
<td>return true if button was up at least once in current frame</td>
</tr>
<tr>
<td>float GetAxisValue (Axis axis) const</td>
<td>get current axis value</td>
</tr>
<tr>
<td>void SetLowFrequencyVibrator (float f)</td>
<td>set low-frequency vibration effect (0.0f .. 1.0f)</td>
</tr>
<tr>
<td>float GetLowFrequencyVibrator () const</td>
<td>get low-frequency vibration</td>
</tr>
<tr>
<td>void SetHighFrequencyVibrator (float f)</td>
<td>set high-frequency vibration effect (0.0f .. 1.0f)</td>
</tr>
<tr>
<td>float GetHighFrequencyVibrator () const</td>
<td>get high-frequency vibration</td>
</tr>
<tr>
<td>bool IsAttached () const</td>
<td>return true if the input handler is currently attached</td>
</tr>
<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 IsCapturing () const</td>
<td>return true if this input handler captures input</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><strong>AddRef</strong> ()</td>
</tr>
<tr>
<td>------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td><em>increment refcount by one</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>Release</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>decrement refcount and destroy object if refcount is zero</em></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><em>return true if this object is instance of given class</em></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><em>return true if this object is instance of given class by string</em></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><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 <strong>Rtti</strong> &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 <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>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>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 OnBeginFrame()</td>
<td>called on InputServer::BeginFrame()</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
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.

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

```cpp
const Util::String & Core::RefCounted::GetClassName
```

get the class name

Get the class name of the object.

```cpp
Util::FourCC Core::RefCounted::GetClassFourCC
```

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.
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><strong>Usage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>resource usage flags</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>State</strong></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><code>IndexBufferBase()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~IndexBufferBase()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>Map(MapType mapType)</code></td>
<td>map index buffer for CPU access</td>
</tr>
<tr>
<td><code>Unmap()</code></td>
<td>unmap the resource</td>
</tr>
</tbody>
</table>

**CoreGraphics::IndexType::Code**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetIndexType()</code></td>
<td>get the index type (Index16 or Index32)</td>
</tr>
<tr>
<td><code>GetNumIndices()</code></td>
<td>get number of indices</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetUsage()</code></td>
<td>get resource usage type</td>
</tr>
</tbody>
</table>

**Access**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 loading</td>
</tr>
<tr>
<td><code>IsAsyncEnabled()</code></td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>SetResourceId(const ResourceId &amp;id)</code></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><code>GetResourceId()</code></td>
<td>get the resource identifier</td>
</tr>
</tbody>
</table>

**Ptr< ResourceLoader >**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetLoader(const Ptr&lt; ResourceLoader &gt; &amp;loader)</code></td>
<td>set optional resource loader</td>
</tr>
<tr>
<td><code>GetLoader()</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>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetSaver</code> () const</td>
<td>set optional resource saver</td>
</tr>
<tr>
<td><code>GetUseCount</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>Unload ()</code></td>
<td>unload the resource, or cancel the pending 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><code>LoadFailed () const</code></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 () 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 a subclass of given class</td>
</tr>
<tr>
<td>return true if this object is instance of given class, or a derived class</td>
<td>bool $\text{IsA}$ (const $\text{Util::String}$ &amp;rttiName) 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>bool $\text{IsA}$ (const $\text{Util::FourCC}$ &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>const $\text{Util::String}$ &amp; $\text{GetClassName}$ () const</td>
</tr>
<tr>
<td>get the class name</td>
<td>$\text{Util::FourCC}$ $\text{GetClassFourCC}$ () 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</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

_dumps 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>void <code>SetIndexType(CoreGraphics::IndexType::Code type)</code></td>
<td>set the index type (Index16 or Index32)</td>
</tr>
<tr>
<td>void <code>SetNumIndices(SizeT num)</code></td>
<td>set number of indices</td>
</tr>
<tr>
<td>void <code>SetUsage(Usage usage)</code></td>
<td>set resource usage type</td>
</tr>
<tr>
<td>void <code>SetAccess(Access access)</code></td>
<td>set resource cpu access type</td>
</tr>
<tr>
<td>void <code>SetState(State s)</code></td>
<td>set current state</td>
</tr>
<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

```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 `CoreGraphics::CPUIndexBuffer`, and `Direct3D9::D3D9IndexBuffer`.

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

unmap the resource

Give up CPU access on the index buffer content.

Reimplemented in `CoreGraphics::CPUIndexBuffer`, and `Direct3D9::D3D9IndexBuffer`.

```cpp
Resource::State Resources::Resource::Load()
```

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 Base::MeshBase, Base::VertexBufferBase, CoreGraphics::CPUIndexBuffer, CoreGraphics::CPUVertexBuffer, Direct3D9::D3D9IndexBuffer, Direct3D9::D3D9Shader, Direct3D9::D3D9Texture, Direct3D9::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.

```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::InputServerBase
#include <inputserverbase.h>

Inheritance diagram for Base::InputServerBase:
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><code>InputServerBase ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~InputServerBase ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <code>Open ()</code></td>
<td>open the input server</td>
</tr>
<tr>
<td>virtual void <code>Close ()</code></td>
<td>close the input server</td>
</tr>
<tr>
<td>bool <code>IsOpen ()</code> const</td>
<td>return true if open</td>
</tr>
<tr>
<td>void <code>SetQuitRequested (bool b)</code></td>
<td>set the quit requested flag</td>
</tr>
<tr>
<td>bool <code>IsQuitRequested ()</code> const</td>
<td>return true if some subsystem has requested to quit the app (e.g. Alt-F4)</td>
</tr>
<tr>
<td>void <code>Reset ()</code></td>
<td>reset input state</td>
</tr>
<tr>
<td>const <code>Ptr&lt;Input::Keyboard&gt; &amp;</code> <code>GetDefaultKeyboard ()</code> const</td>
<td>get the default keyboard input handler</td>
</tr>
<tr>
<td>const <code>Ptr&lt;Input::Mouse&gt; &amp;</code> <code>GetDefaultMouse ()</code> const</td>
<td>get the default mouse input handler</td>
</tr>
<tr>
<td>const <code>Ptr&lt;Input::GamePad&gt; &amp;</code> <code>GetDefaultGamePad (IndexT playerIndex)</code> const</td>
<td>get default gamepad handler (up to 4)</td>
</tr>
<tr>
<td>void <code>AttachInputHandler</code></td>
<td>attach an input handler</td>
</tr>
<tr>
<td>void <code>RemoveInputHandler</code></td>
<td>remove an input handler</td>
</tr>
<tr>
<td>virtual void <code>BeginFrame ()</code></td>
<td>call before processing window events</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>virtual void <strong>OnFrame</strong> ()</td>
<td>call after processing window events</td>
</tr>
<tr>
<td>void <strong>EndFrame</strong> ()</td>
<td>call at end of frame</td>
</tr>
<tr>
<td>void <strong>PutEvent</strong> (const <strong>Input::InputEvent</strong> &amp;ie)</td>
<td>put an input event into the handler chain</td>
</tr>
<tr>
<td>void <strong>ClearMouseCapture</strong> ()</td>
<td>clear the current mouse capture (if exists)</td>
</tr>
<tr>
<td>void <strong>ClearKeyboardCapture</strong> ()</td>
<td>clear the current keyboard capture (if exists)</td>
</tr>
<tr>
<td>void <strong>ClearCapture</strong> ()</td>
<td>clear both mouse and keyboard captures</td>
</tr>
</tbody>
</table>

**GetMouseCaptureHandler** () const
const **Ptr** < **Input::InputHandler** > &
return the current mouse capture input handler (return invalid ptr if no capture set)

**GetKeyboardCaptureHandler** () const
const **Ptr** < **Input::InputHandler** > &
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

void **ReleaseKeyboardCapture** (const **Ptr**< **Input::InputHandler** > &inputHandler)
only call from InputHandler: release keyboard capture

int **GetRefCount** () const
get the current refcount

void **AddRef** ()
increment refcount by one

void **Release** ()
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 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>
Member Function Documentation

void Base::InputServerBase::EndFrame() 

call at end of frame

Call this somewhere towards the end of frame, when it is guaranteed 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.

const Ptr<
void Base::InputServerBase::ObtainMouseCapture(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.

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::KeyboardBase
Base::KeyboardBase Class Reference

#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 FourCC</td>
</tr>
</tbody>
</table>
`return true if this object is instance of given class by fourcc`  

<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><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>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

```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::MemoryIndexBufferLoaderBase**
#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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemoryIndexBufferLoaderBase ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>Setup (CoreGraphics::IndexType::Code indexType, SizeT numIndices, void *ptr, SizeT numBytes)</td>
<td>setup index buffer data, must remain valid until OnLoadRequested() is called!</td>
</tr>
<tr>
<td>Setup (CoreGraphics::IndexType::Code indexType, SizeT numIndices, SizeT numBytes, CoreGraphics::IndexBuffer::Usage usage, CoreGraphics::IndexBuffer::Access access)</td>
<td>setup a empty index buffer</td>
</tr>
<tr>
<td>Setup (CoreGraphics::IndexType::Code type, SizeT num, void *ptr, SizeT numBytes, CoreGraphics::IndexBuffer::Usage usage, CoreGraphics::IndexBuffer::Access access)</td>
<td>setup index buffer data, must remain valid until OnLoadRequested() is called!</td>
</tr>
<tr>
<td>OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</td>
<td>called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>OnRemoveFromResource ()</td>
<td>called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>IsAttachedToResource () const</td>
<td>return true if attached to resource</td>
</tr>
<tr>
<td>GetResource () const</td>
<td>get pointer to resource</td>
</tr>
<tr>
<td>CanLoadAsync () const</td>
<td>return true if asynchronous loading is supported</td>
</tr>
<tr>
<td>OnLoadRequested ()</td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td>OnLoadCancelled ()</td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td>OnPending ()</td>
<td></td>
</tr>
</tbody>
</table>
Call frequently while after `OnLoadRequested()` to put `Resource` into loaded state

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Resource::State GetState () const</code></td>
<td>return current 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><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

```c
void SetState (Resource::State S)
set current state
```
Member Function Documentation

void Base::MemoryIndexBufferLoaderBase::Setup (CoreGraphics::IndexType::Code indexType,
    SizeT num,
    void * ptr,
    SizeT numBytes)

setup index buffer data, must remain valid until **OnLoadRequested()** is called!

Setup all information needed to initialize the IndexBuffer resource. The data must remain valid until **OnLoadRequested()** is called (which will invalidate the data).

void Base::MemoryIndexBufferLoaderBase::Setup (CoreGraphics::IndexType::Code indexType,
    SizeT num,
    SizeT numBytes,
    CoreGraphics::IndexBuffer::Usage usage,
    CoreGraphics::IndexBuffer::Access access)

setup a empty index buffer

Setup all information needed to initialize a empty IndexBuffer resource.

void Base::MemoryIndexBufferLoaderBase::Setup (CoreGraphics::IndexType::Code type,
    SizeT num,
    void * ptr,
    SizeT numBytes,
    CoreGraphics::IndexBuffer::Usage usage,
    CoreGraphics::IndexBuffer::Access access)

setup index buffer data, must remain valid until **OnLoadRequested()** is called!
Setup all information needed to initialize a IndexBuffer resource.

```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`, `Direct3D9::D3D9StreamTextureLoader`, `CoreGraphics::StreamAnimationLoader`, `CoreGraphics::StreamMeshLoader`, and `Models::StreamModelLoader`.

```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::CPUMemoryIndexBufferLoader`, `CoreGraphics::CPUMemoryVertexBufferLoader`, `Direct3D9::D3D9MemoryIndexBufferLoader`, `Direct3D9::D3D9MemoryVertexBufferLoader`, `Direct3D9::D3D9StreamShaderLoader`, `Direct3D9::D3D9StreamTextureLoader`, `CoreGraphics::StreamAnimationLoader`. 
CoreGraphics::StreamMeshLoader, Models::StreamModelLoader, and Resources::DynamicMeshResourceLoader.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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::MemoryVertexBufferLoaderBase
#include <memoryvertexbufferloaderbase.h>

Inheritance diagram for Base::MemoryVertexBufferLoaderBase:
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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemoryVertexBufferLoaderBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>*<em>void Setup (const Util::Array&lt; CoreGraphics::VertexComponent &gt; &amp;vertexComponents, SizeT numVertices, void <em>ptr, SizeT numBytes)</em></em></td>
<td>setup vertex buffer data, must remain valid until OnLoadRequested() is called!</td>
</tr>
<tr>
<td><strong>void Setup (const Util::Array&lt; CoreGraphics::VertexComponent &gt; &amp;vertexComponents, SizeT numVertices, SizeT numBytes, CoreGraphics::VertexBuffer::Usage usage, CoreGraphics::VertexBuffer::Access access)</strong></td>
<td>setup a empty vertex buffer</td>
</tr>
<tr>
<td>*<em>void Setup (const Util::Array&lt; CoreGraphics::VertexComponent &gt; &amp;vertexComponents, SizeT numVertices, void <em>ptr, SizeT numBytes, CoreGraphics::VertexBuffer::Usage usage, CoreGraphics::VertexBuffer::Access access)</em></em></td>
<td>setup a vertex buffer, vertex buffer data, must remain valid until OnLoadRequested() is called!</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; 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>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>virtual bool</strong> OnLoadRequested ()</td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td><strong>virtual void</strong> OnLoadCancelled ()</td>
<td>called by resource to cancel a pending load</td>
</tr>
<tr>
<td><strong>virtual bool</strong> OnPending ()</td>
<td>call frequently while after OnLoadRequested() to put Resource into loaded state</td>
</tr>
<tr>
<td><strong>Resource::State</strong> GetState () const</td>
<td>return current state</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 Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool</strong> 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><strong>bool</strong> 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><strong>bool</strong> 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><strong>bool</strong> 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><strong>bool</strong> 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><strong>const Util::String &amp;</strong> GetClassName () const</td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong> 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

void SetState (Resource::State S)
set current state
Member Function Documentation

void Base::MemoryVertexBufferLoaderBase::Setup(
  const Util::Array<
    CoreGraphics::VertexComponent
  > & components,
  SizeT num,
  void * ptr,
  SizeT numBytes
)

setup vertex buffer data, must remain valid until OnLoadRequested() is called!

Setup all information needed to initialize the VertexBuffer resource. The data must remain valid until OnLoadRequested() is called (which will invalidate the data).

void Base::MemoryVertexBufferLoaderBase::Setup(
  const Util::Array<
    CoreGraphics::VertexComponent
  > & vertexComponents,
  SizeT num,
  SizeT numBytes
)

setup a empty vertex buffer

Setup all information needed to initialize a empty VertexBuffer resource.

void Base::MemoryVertexBufferLoaderBase::Setup(
  const Util::Array<
    CoreGraphics::VertexComponent
  > & components,
  SizeT num,
  void * ptr,
  SizeT numBytes
)

    CoreGraphics::VertexBuffer::Usage
  usage,
  CoreGraphics::VertexBuffer::Access
  access
)

setup a vertex buffer, vertex buffer data, must remain valid until
`OnLoadRequested()` is called!

Setup all information needed to initialize the VertexBuffer resource. The data must remain valid until `OnLoadRequested()` is called (which will invalidate the data).

```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`, `Direct3D9::D3D9StreamTextureLoader`, `CoreGraphics::StreamAnimationLoader`, `CoreGraphics::StreamMeshLoader`, and `Models::StreamModelLoader`.

```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::CPUMemoryIndexBufferLoader`, `CoreGraphics::CPUMemoryVertexBufferLoader`, `CoreGraphics::CPUMemoryVertexBufferLoader`, and `CoreGraphics::CPUMemoryVertexBufferLoader`. 
Direct3D9::D3D9MemoryIndexBufferLoader,
Direct3D9::D3D9MemoryVertexBufferLoader,
Direct3D9::D3D9StreamShaderLoader,
Direct3D9::D3D9StreamTextureLoader,
CoreGraphics::StreamAnimationLoader,
CoreGraphics::StreamMeshLoader, Models::StreamModelLoader, and Resources::DynamicMeshResourceLoader.

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 Direct3D9::D3D9StreamTextureLoader,
CoreGraphics::StreamAnimationLoader,
CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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 Direct3D9::D3D9StreamTextureLoader,
CoreGraphics::StreamAnimationLoader,
CoreGraphics::StreamMeshLoader, and
Models::StreamModelLoader.

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::MeshBase
Base::MeshBase Class Reference

#include <meshbase.h>

Inheritance diagram for Base::MeshBase:
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
### 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>MeshBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~MeshBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual void Unload ()</strong></td>
<td>unload mesh resource</td>
</tr>
<tr>
<td><strong>bool HasVertexBuffer () const</strong></td>
<td>return true if the mesh has a vertex buffer</td>
</tr>
<tr>
<td><strong>const Ptr &lt; CoreGraphics::VertexBuffer &gt; &amp; GetVertexBuffer () const</strong></td>
<td>get the vertex buffer object</td>
</tr>
<tr>
<td><strong>bool HasIndexBuffer () const</strong></td>
<td>return true if the mesh has an index buffer</td>
</tr>
<tr>
<td><strong>const Ptr &lt; CoreGraphics::IndexBuffer &gt; &amp; GetIndexBuffer () const</strong></td>
<td>get the index buffer object</td>
</tr>
<tr>
<td><strong>SizeT GetNumPrimitiveGroups () const</strong></td>
<td>get the number of primitive groups in the mesh</td>
</tr>
<tr>
<td><strong>const CoreGraphics::PrimitiveGroup &amp; GetPrimitiveGroupAtIndex (IndexT i) const</strong></td>
<td>get primitive group at index</td>
</tr>
<tr>
<td><strong>virtual void ApplyPrimitives (IndexT primGroupIndex)</strong></td>
<td>apply any necessary mesh data in renderdevice</td>
</tr>
<tr>
<td><strong>void SetPrimitiveGroups (const Util::Array&lt; CoreGraphics::PrimitiveGroup &gt; &amp;groups)</strong></td>
<td>set primitive groups</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 SetResourceId (const ResourceId &amp;id)</strong></td>
<td>set resource id</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>const ResourceId &amp; GetResourceId() const</td>
<td>set the resource identifier</td>
</tr>
<tr>
<td>get the resource identifier</td>
<td></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>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>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 current state is Failed</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 (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 <strong>SetVertexBuffer</strong> (const Ptr&lt; CoreGraphics::VertexBuffer &gt; &amp;vb)</td>
<td>set the vertex buffer object</td>
</tr>
<tr>
<td>void <strong>SetIndexBuffer</strong> (const Ptr&lt; CoreGraphics::IndexBuffer &gt; &amp;ib)</td>
<td>set the index buffer object</td>
</tr>
<tr>
<td>void <strong>setState</strong> (State s)</td>
<td>set current state</td>
</tr>
<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
class 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::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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MouseBase ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~MouseBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual void BeginCapture ()</strong></td>
<td>capture input to this event handler</td>
</tr>
<tr>
<td><strong>virtual void EndCapture ()</strong></td>
<td>end input capturing to this event handler</td>
</tr>
<tr>
<td><strong>bool ButtonPressed (Input::MouseButton::Code btn) const</strong></td>
<td>return true if button is currently pressed</td>
</tr>
<tr>
<td><strong>bool ButtonDown (Input::MouseButton::Code btn) const</strong></td>
<td>return true if button was down at least once in current frame</td>
</tr>
<tr>
<td><strong>bool ButtonUp (Input::MouseButton::Code btn) const</strong></td>
<td>return true if button was up at least once in current frame</td>
</tr>
<tr>
<td><strong>bool ButtonDoubleClicked (Input::MouseButton::Code btn) const</strong></td>
<td>return true if a button has been double clicked</td>
</tr>
<tr>
<td><strong>bool WheelForward () const</strong></td>
<td>return true if mouse wheel rotated forward</td>
</tr>
<tr>
<td><strong>bool WheelBackward () const</strong></td>
<td>return true if mouse wheel rotated backward</td>
</tr>
<tr>
<td><strong>const Math::float2 &amp; GetPixelPosition () const</strong></td>
<td>get current absolute mouse position (in pixels)</td>
</tr>
<tr>
<td><strong>const Math::float2 &amp; GetScreenPosition () const</strong></td>
<td>get current screen space mouse position (0.0 .. 1.0)</td>
</tr>
<tr>
<td><strong>const Math::float2 &amp; GetMovement () const</strong></td>
<td>get mouse movement</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>bool IsCapturing () const</strong></td>
<td>return true if this input handler captures input</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 & 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 (NEBULA3_DEBUG builds only!)
```
### Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void</td>
<td><strong>OnAttach</strong> ()</td>
<td>called when the handler is attached to the input server</td>
</tr>
<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

```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::PreShader
Base::PreShader Class Reference

#include <preshader.h>
Detailed Description

A PreShader is attached to a shader instance and will be called back to setup shader variables before the shader is "rendered".

(C) 2007 Radon Labs GmbH
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

**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><code>RenderDeviceBase()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>virtual ~RenderDeviceBase()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>Open the device</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>Close the device</td>
</tr>
<tr>
<td><code>IsOpen()</code> const</td>
<td>Return true if currently open</td>
</tr>
<tr>
<td><code>AttachEventHandler(const Ptr&lt;CoreGraphics::RenderEventHandler&gt; &amp;h)</code></td>
<td>Attach a render event handler</td>
</tr>
<tr>
<td><code>RemoveEventHandler(const Ptr&lt;CoreGraphics::RenderEventHandler&gt; &amp;h)</code></td>
<td>Remove a render event handler</td>
</tr>
<tr>
<td><code>GetDefaultRenderTarget()</code> const</td>
<td>Get default render target</td>
</tr>
<tr>
<td><code>BeginFrame()</code></td>
<td>Begin complete frame</td>
</tr>
<tr>
<td><code>BeginPass(const Ptr&lt;CoreGraphics::RenderTarget&gt; &amp;rt, const Ptr&lt;CoreGraphics::ShaderInstance&gt; &amp;passShader)</code></td>
<td>Begin rendering a frame pass</td>
</tr>
<tr>
<td><code>BeginBatch(CoreGraphics::BatchType::Code batchType, const Ptr&lt;CoreGraphics::ShaderInstance&gt; &amp;batchShader)</code></td>
<td>Begin batch rendering</td>
</tr>
</tbody>
</table>

---
void SetVertexBuffer (const Ptr\< CoreGraphics::VertexBuffer \>& vb)  
set current vertex buffer

const Ptr\< CoreGraphics::VertexBuffer \>& GetVertexBuffer () const  
get current vertex buffer

void SetIndexBuffer (const Ptr\< CoreGraphics::IndexBuffer \>& ib)  
set current index buffer

const Ptr\< CoreGraphics::IndexBuffer \>& GetIndexBuffer () const  
get current index buffer

void SetPrimitiveGroup (const CoreGraphics::PrimitiveGroup & pg)  
set current primitive group

const CoreGraphics::PrimitiveGroup & GetPrimitiveGroup () const  
get current primitive group

void Draw ()  
draw current primitives

void EndBatch ()  
end current batch

void EndPass ()  
end current pass

void EndFrame ()  
end current frame

bool IsInBeginFrame () const  
check if inside BeginFrame

void Present ()  
present the rendered scene

void SaveScreenshot (CoreGraphics::ImageFileFormat::Code fmt, const Ptr\< IO::Stream \>& outStream)  
save a screenshot to the provided stream

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, 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 bool</th>
<th>CanCreate ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>test if a compatible render device can be created on this machine</em></td>
<td></td>
</tr>
</tbody>
</table>

<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

```cpp
bool NotifyEventHandlers (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::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.
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::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
## 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><code>~RenderTargetBase()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>IsDefaultRenderTarget()</code> const</td>
<td>Get default render target flag</td>
</tr>
<tr>
<td><code>HasColorBuffer(IndexT colorBufferIndex)</code></td>
<td>Return true if color buffer exists</td>
</tr>
<tr>
<td><code>HasDepthStencilBuffer()</code> const</td>
<td>Return true if the render target has a depth/stencil buffer</td>
</tr>
<tr>
<td><code>IsValid()</code> const</td>
<td>Return true if valid (has been setup)</td>
</tr>
<tr>
<td><code>setWidth(SizeT w)</code></td>
<td>Set render target width</td>
</tr>
<tr>
<td><code>GetWidth()</code> const</td>
<td>Get width of render target in pixels</td>
</tr>
<tr>
<td><code>SetHeight(SizeT h)</code></td>
<td>Set render target height</td>
</tr>
<tr>
<td><code>GetHeight()</code> const</td>
<td>Get height of render target in pixels</td>
</tr>
<tr>
<td><code>SetAntiAliasQuality(CoreGraphics::AntiAliasQuality::Code c)</code></td>
<td>Set antialias quality</td>
</tr>
<tr>
<td><code>GetAntiAliasQuality()</code> const</td>
<td>Get anti-alias-quality</td>
</tr>
<tr>
<td><code>AddColorBuffer(CoreGraphics::PixelFormat::Code colorFormat)</code></td>
<td>Add a color buffer</td>
</tr>
<tr>
<td><code>GetNumColorBuffers()</code> const</td>
<td>Get number of color buffers</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetColorBufferFormat</code></td>
<td>Get color buffer format at index</td>
</tr>
<tr>
<td><code>AddDepthStencilBuffer</code></td>
<td>Add a depth/stencil buffer</td>
</tr>
<tr>
<td><code>SetMipMapsEnabled</code></td>
<td>Enable mipmap generation for this render target</td>
</tr>
<tr>
<td><code>AreMipMapsEnabled</code></td>
<td>Get mipmap generation flag</td>
</tr>
<tr>
<td><code>SetResolveTextureResourceId</code></td>
<td>Set resolve texture resource id</td>
</tr>
<tr>
<td><code>GetResolveTextureResourceId</code></td>
<td>Get resolve texture resource id</td>
</tr>
<tr>
<td><code>SetResolveTextureWidth</code></td>
<td>Set resolve texture width</td>
</tr>
<tr>
<td><code>GetResolveTextureWidth</code></td>
<td>Get resolve texture width</td>
</tr>
<tr>
<td><code>SetResolveTextureHeight</code></td>
<td>Set resolve texture height</td>
</tr>
<tr>
<td><code>GetResolveTextureHeight</code></td>
<td>Get resolve texture height</td>
</tr>
<tr>
<td><code>SetClearColor</code></td>
<td>Set clear color</td>
</tr>
<tr>
<td><code>GetClearColor</code></td>
<td>Get clear color</td>
</tr>
<tr>
<td><code>SetClearDepth</code></td>
<td>Set clear depth</td>
</tr>
<tr>
<td><code>GetClearDepth</code></td>
<td>Get clear depth</td>
</tr>
<tr>
<td><code>SetClearStencil</code></td>
<td>Set clear stencil value</td>
</tr>
<tr>
<td><code>GetClearStencil</code></td>
<td>Get clear stencil value</td>
</tr>
<tr>
<td><code>SetResolveRect</code></td>
<td>Set resolve rect</td>
</tr>
<tr>
<td><code>GetResolveRect</code></td>
<td>Get resolve rect</td>
</tr>
</tbody>
</table>
set the current resolve rectangle (in pixels)

const Math::rectangle< uint >& GetResolveRect() const

get resolve rectangle

void Setup()

setup the render target object

void Discard()

discard the render target object

void BeginPass()

begin rendering to the render target

void BeginBatch(CoreGraphics::BatchType::Code batchType)

begin a batch

void EndBatch()

end current batch

void EndPass()

end current render pass

void GenerateMipLevels()

generate mipmap levels

bool HasResolveTexture() const

return true if resolve texture is valid

const Ptr< CoreGraphics::Texture >& GetResolveTexture() const

get the resolve texture as Nebula texture object

int GetRefCount()

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
<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>
<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>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>
Static Public Attributes

<table>
<thead>
<tr>
<th>static const IndexT</th>
<th>MaxNumColorBuffers</th>
<th>4</th>
</tr>
</thead>
</table>

max number of color buffers
Protected Member Functions

void SetDefaultRenderTarget (bool b)

set to true if default render target
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::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><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><code>ResourceBase()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~ResourceBase()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>GetUsage()</code> const</td>
<td>get resource usage type</td>
</tr>
<tr>
<td><code>GetAccess()</code> const</td>
<td>get cpu access type</td>
</tr>
<tr>
<td><code>SetAsyncEnabled(bool b)</code></td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td><code>IsAsyncEnabled()</code> const</td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>SetResourceId(const ResourceId &amp;id)</code></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><code>GetResourceId()</code> const</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()</code> const</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()</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>Unload()</code></td>
<td>unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td><code>GetState()</code> const</td>
<td>get state</td>
</tr>
</tbody>
</table>
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

<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>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetUsage (Usage usage)</code></td>
<td>set 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>void SetState (State s)</code></td>
<td>set current state</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

**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 **Base::MeshBase**, **Base::VertexBufferBase**, **CoreGraphics::CPUIndexBuffer**, **CoreGraphics::CPUVertexBuffer**, **Direct3D9::D3D9IndexBuffer**, **Direct3D9::D3D9Shader**, **Direct3D9::D3D9Texture**, **Direct3D9::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.
```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

Base::ShaderBase
#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

enum **State**

resource states *(DO NOT CHANGE ORDER!)*
## Public Member Functions

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Constructor</td>
<td>ShaderBase ()</td>
<td>create a shader instance from this shader</td>
</tr>
<tr>
<td>Destructor</td>
<td>~ShaderBase ()</td>
<td>discard a shader instance</td>
</tr>
<tr>
<td>Function</td>
<td>Ptr<a href="">CoreGraphics::ShaderInstance</a> CreateShaderInstance ()</td>
<td>create a shader instance from this shader</td>
</tr>
<tr>
<td>Function</td>
<td>void DiscardShaderInstance (const Ptr<a href="">CoreGraphics::ShaderInstance</a>&amp; inst)</td>
<td>discard a shader instance</td>
</tr>
<tr>
<td>Function</td>
<td>const Util::Array&lt;Ptr/CoreGraphics::ShaderInstance&gt;&amp; GetAllShaderInstances () const</td>
<td>get all instances</td>
</tr>
<tr>
<td>Function</td>
<td>void SetAsyncEnabled (bool b)</td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td>Function</td>
<td>bool IsAsyncEnabled () const</td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td>Function</td>
<td>void SetResourceld (const Resourceld&amp;id)</td>
<td>set the resource identifier</td>
</tr>
<tr>
<td>Function</td>
<td>const Resourceld &amp; GetResourceld () const</td>
<td>get the resource identifier</td>
</tr>
<tr>
<td>Function</td>
<td>void SetLoader (const Ptr&lt;ResourceLoader&gt;&amp; loader)</td>
<td>set optional resource loader</td>
</tr>
<tr>
<td>Function</td>
<td>const Ptr&lt;ResourceLoader&gt;&amp; GetLoader () const</td>
<td>get optional resource loader</td>
</tr>
<tr>
<td>Function</td>
<td>void SetSaver (const Ptr&lt;ResourceSaver&gt;&amp; saver)</td>
<td>set optional resource saver</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>GetSaver()</code> const</td>
<td>gets optional resource saver</td>
<td></td>
</tr>
<tr>
<td><code>GetUseCount()</code> const</td>
<td>gets current use count</td>
<td></td>
</tr>
<tr>
<td><code>Load()</code></td>
<td>load the resource</td>
<td></td>
</tr>
<tr>
<td><code>Unload()</code></td>
<td>unload the resource, or cancel the pending load</td>
<td></td>
</tr>
<tr>
<td><code>GetState()</code> const</td>
<td>gets current state</td>
<td></td>
</tr>
<tr>
<td><code>IsLoaded()</code> const</td>
<td>return true if current state is Loaded</td>
<td></td>
</tr>
<tr>
<td><code>IsPending()</code> const</td>
<td>return true if current state is Pending</td>
<td></td>
</tr>
<tr>
<td><code>LoadFailed()</code> const</td>
<td>return true if current state is Failed</td>
<td></td>
</tr>
<tr>
<td><code>Save()</code></td>
<td>save the resource</td>
<td></td>
</tr>
<tr>
<td><code>GetRefCount()</code> const</td>
<td>gets the current refcount</td>
<td></td>
</tr>
<tr>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
<td></td>
</tr>
<tr>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
<td></td>
</tr>
<tr>
<td><code>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>IsInstanceOf(const Util::String &amp;className)</code></td>
<td>return true if this object is instance of given class by string</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>
<td></td>
</tr>
<tr>
<td><code>IsA(const Rtti &amp;rtti)</code> const</td>
<td>return true if this object is instance of given class by Rtti</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

getc the class name

Util::FourCC GetClassFourCC () const

getc 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

- void **SetState** *(State S)*
  - set current state
- void **IncrUseCount** *
  - increment use count
- void **DecrUseCount** *
  - decrement use count
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 `Base::MeshBase`, `Base::VertexBufferBase`, `CoreGraphics::CPUIndexBuffer`, `CoreGraphics::CPUVertexBuffer`, `Direct3D9::D3D9IndexBuffer`, `Direct3D9::D3D9Shader`, `Direct3D9::D3D9Texture`, `Direct3D9::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.
Base::ShaderInstanceBase
#include <shaderinstancebase.h>

Inheritance diagram for Base::ShaderInstanceBase:
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><code>ShaderInstanceBase()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>virtual ~ShaderInstanceBase()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>Discard()</code></td>
<td><code>discard the shader instance, must be called when instance no longer needed</code></td>
</tr>
<tr>
<td><code>IsValid()</code></td>
<td><code>return true if this object is valid</code></td>
</tr>
<tr>
<td><code>GetOriginalShader()</code></td>
<td><code>get pointer to original shader which created this instance</code></td>
</tr>
<tr>
<td><code>HasVariableByName(const CoreGraphics::ShaderVariable::Name &amp;n)</code></td>
<td><code>return true if the shader instance has a variable by name</code></td>
</tr>
<tr>
<td><code>HasVariableBySemantic(const CoreGraphics::ShaderVariable::Semantic &amp;s)</code></td>
<td><code>return true if shader has variable by semantic</code></td>
</tr>
<tr>
<td><code>GetNumVariables()</code></td>
<td><code>get number of variables</code></td>
</tr>
<tr>
<td><code>GetVariableByIndex(IndexT i)</code></td>
<td><code>get a variable by index</code></td>
</tr>
<tr>
<td><code>GetVariableByName(const CoreGraphics::ShaderVariable::Name &amp;n)</code></td>
<td><code>get a variable by name</code></td>
</tr>
<tr>
<td><code>GetVariableBySemantic(const CoreGraphics::ShaderVariable::Semantic &amp;s)</code></td>
<td><code>get a variable by semantic</code></td>
</tr>
<tr>
<td><code>HasVariation(CoreGraphics::ShaderFeature::Mask)</code></td>
<td></td>
</tr>
</tbody>
</table>

---

`public`, `member`, `functions`, `ShaderInstanceBase`, `constructor`, `virtual`, `destructor`, `void`, `Discard`, `bool`, `IsValid`, `const`, `Ptr`, `CoreGraphics::Shader`, `GetOriginalShader`, `HasVariableByName`, `CoreGraphics::ShaderVariable::Name`, `HasVariableBySemantic`, `CoreGraphics::ShaderVariable::Semantic`, `GetNumVariables`, `GetVariableByIndex`, `GetVariableByName`, `GetVariableBySemantic`, `HasVariation`, `CoreGraphics::ShaderFeature::Mask`
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SizeT GetNumVariations()</code> const</td>
<td>get number of variations in the shader</td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::ShaderVariation&gt; &amp; GetVariationByIndex(IndexT i)</code> const</td>
<td>get shader variation by index</td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::ShaderVariation&gt; &amp; GetVariationByFeatureMask(CoreGraphics::ShaderFeature::Mask featureMask)</code> const</td>
<td>get shader variation by feature mask</td>
</tr>
<tr>
<td><code>bool SelectActiveVariation(CoreGraphics::ShaderFeature::Mask featureMask)</code></td>
<td>select active variation by feature mask, return true if active variation has been changed</td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::ShaderVariation&gt; &amp; GetActiveVariation()</code> const</td>
<td>get currently active variation</td>
</tr>
<tr>
<td><code>void AddPreShader(const Ptr&lt;CoreGraphics::PreShader&gt; &amp;preShader)</code></td>
<td>add a pre-shader</td>
</tr>
<tr>
<td><code>void RemovePreShader(const Ptr&lt;CoreGraphics::PreShader&gt; &amp;preShader)</code></td>
<td>remove a pre-shader</td>
</tr>
<tr>
<td><code>const Util::Array&lt;Ptr&lt;CoreGraphics::PreShader&gt;&gt; &amp; GetPreShaders()</code> const</td>
<td>get array of pre-shaders</td>
</tr>
<tr>
<td><code>void Begin()</code></td>
<td>begin rendering through the currently selected variation, returns no. passes (for rendering)</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 (for 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>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</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><strong>bool IsInstanceOf (const Rtti &amp;rtti)</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)</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)</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)</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 ()</strong></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC GetClassFourCC ()</strong></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 <strong>Setup</strong> (const Ptr&lt; CoreGraphics::Shader &gt; &amp;origShader)</td>
<td>setup the shader instance from its original shader object</td>
</tr>
<tr>
<td>virtual void <strong>Cleanup</strong> ()</td>
<td>cleanup the shader instance</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()

cleanup 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.

```
void 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::ShaderServerBase
Base::ShaderServerBase Class Reference

#include <shaderserverbase.h>

Inheritance diagram for Base::ShaderServerBase:

- Core::RefCounted
- Base::ShaderServerBase
- Direct3D9::D3D9ShaderServer
- CoreGraphics::ShaderServer
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 Types

<table>
<thead>
<tr>
<th>enum</th>
<th>ShaderParamBindMode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shader parameter bind modes</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ShaderServerBase()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~ShaderServerBase()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>SetShaderParamBindMode(ShaderParamBindMode m)</code></td>
<td>Set shader param bind mode (by name or by semantic, default is by semantic)</td>
</tr>
<tr>
<td><code>GetShaderParamBindMode()</code> const</td>
<td>Get shader param bind mode</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>Open the shader server</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>Close the shader server</td>
</tr>
<tr>
<td><code>IsOpen()</code> const</td>
<td>Return true if the shader server is open</td>
</tr>
<tr>
<td><code>HasShader(const Resources::ResourceId &amp;resId)</code> const</td>
<td>Return true if a shader exists</td>
</tr>
<tr>
<td><code>CreateShaderInstance(const Resources::ResourceId &amp;resId)</code></td>
<td>Create a new shader instance</td>
</tr>
<tr>
<td><code>GetAllShaders()</code> const</td>
<td>Get all loaded shaders</td>
</tr>
<tr>
<td><code>SetActiveShaderInstance(const Ptr&lt;CoreGraphics::ShaderInstance&gt; &amp;shaderInst)</code></td>
<td>Set currently active shader instance</td>
</tr>
<tr>
<td><code>GetActiveShaderInstance()</code> const</td>
<td>Get currently active shader instance</td>
</tr>
<tr>
<td><code>ResetFeatureBits()</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void SetFeatureBits(CoreGraphics::ShaderFeature::Mask m)</code></td>
<td>set shader feature by bit mask</td>
</tr>
<tr>
<td><code>void ClearFeatureBits(CoreGraphics::ShaderFeature::Mask m)</code></td>
<td>clear shader feature by bit mask</td>
</tr>
<tr>
<td><code>CoreGraphics::ShaderFeature::Mask GetFeatureBits()</code></td>
<td>get the current feature mask</td>
</tr>
<tr>
<td><code>CoreGraphics::ShaderFeature::Mask FeatureStringToMask(const Util::String &amp;str)</code></td>
<td>convert a shader feature string into a feature bit mask</td>
</tr>
<tr>
<td><code>Util::String FeatureMaskToString(CoreGraphics::ShaderFeature::Mask mask)</code></td>
<td>convert shader feature bit mask into string</td>
</tr>
<tr>
<td><code>bool HasSharedVariableByName(const CoreGraphics::ShaderVariable::Name &amp;name) const</code></td>
<td>return true if a shared variable exists by name</td>
</tr>
<tr>
<td><code>bool HasSharedVariableBySemantic(const CoreGraphics::ShaderVariable::Semantic &amp;sem) const</code></td>
<td>return true if a shared variable exists by semantic</td>
</tr>
<tr>
<td><code>SizeT GetNumSharedVariables()</code></td>
<td>get number of shared variables</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp; GetSharedVariableByIndex(IndexT i) const</code></td>
<td>get a shared variable by index</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp; GetSharedVariableByName(const CoreGraphics::ShaderVariable::Name &amp;name) const</code></td>
<td>get a shared variable by name</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp; GetSharedVariableBySemantic(const CoreGraphics::ShaderVariable::Semantic &amp;sem) const</code></td>
<td>get a shared variable by semantic</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, 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

```plaintext
static void DumpRefCountingLeaks ()

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

dump 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::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><strong>Type</strong></td>
<td>shader variable types</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>shader variable name typedef</td>
</tr>
<tr>
<td><strong>Semantic</strong></td>
<td>shader variable semantic typedef</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></td>
</tr>
<tr>
<td><em>virtual</em></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~ShaderVariableBase ()</strong></td>
<td></td>
</tr>
<tr>
<td><em>destructor</em></td>
<td></td>
</tr>
<tr>
<td><strong>Ptr &lt; CoreGraphics::ShaderVariableInstance &gt; CreateInstance ()</strong></td>
<td></td>
</tr>
<tr>
<td><em>create a shader variable instance</em></td>
<td></td>
</tr>
<tr>
<td><strong>Type GetType () const</strong></td>
<td></td>
</tr>
<tr>
<td><em>get the data type of the variable</em></td>
<td></td>
</tr>
<tr>
<td><strong>const Name &amp; GetName () const</strong></td>
<td></td>
</tr>
<tr>
<td><em>get the name of the variable</em></td>
<td></td>
</tr>
<tr>
<td><strong>const Semantic &amp; GetSemantic () const</strong></td>
<td></td>
</tr>
<tr>
<td><em>get the semantics of the variable</em></td>
<td></td>
</tr>
<tr>
<td><strong>SizeT GetNumArrayElements () const</strong></td>
<td></td>
</tr>
<tr>
<td><em>number of array elements if this is an array variable</em></td>
<td></td>
</tr>
<tr>
<td><strong>bool IsArray () const</strong></td>
<td></td>
</tr>
<tr>
<td><em>return true if this is an array</em></td>
<td></td>
</tr>
<tr>
<td><strong>void SetInt (int value)</strong></td>
<td></td>
</tr>
<tr>
<td><em>set int value</em></td>
<td></td>
</tr>
<tr>
<td>*<em>void SetIntArray (const int <em>values, SizeT count)</em></em></td>
<td></td>
</tr>
<tr>
<td><em>set int array values</em></td>
<td></td>
</tr>
<tr>
<td><strong>void SetFloat (float value)</strong></td>
<td></td>
</tr>
<tr>
<td><em>set float value</em></td>
<td></td>
</tr>
<tr>
<td>*<em>void SetFloatArray (const float <em>values, SizeT count)</em></em></td>
<td></td>
</tr>
<tr>
<td><em>set float array values</em></td>
<td></td>
</tr>
<tr>
<td><strong>void SetVector (const Math::float4 &amp;value)</strong></td>
<td></td>
</tr>
<tr>
<td><em>set vector value</em></td>
<td></td>
</tr>
<tr>
<td>*<em>void SetVectorArray (const Math::float4 <em>values, SizeT count)</em></em></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>SetMatrix</code></td>
<td>Sets the matrix array values.</td>
</tr>
<tr>
<td><code>SetMatrixArray</code></td>
<td>Sets the matrix array values.</td>
</tr>
<tr>
<td><code>SetBool</code></td>
<td>Sets the bool array values.</td>
</tr>
<tr>
<td><code>SetBoolArray</code></td>
<td>Sets the bool array values.</td>
</tr>
<tr>
<td><code>SetTexture</code></td>
<td>Sets the texture value.</td>
</tr>
<tr>
<td><code>GetRefCount</code></td>
<td>Returns the current refcount.</td>
</tr>
<tr>
<td><code>AddRef</code></td>
<td>Increments the refcount by one.</td>
</tr>
<tr>
<td><code>Release</code></td>
<td>Decrements the refcount and destroys the object if refcount is zero.</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>Returns true if this object is an instance of the given class.</td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>Returns true if this object is an instance of the given class.</td>
</tr>
<tr>
<td>Type</td>
<td>Method</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>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>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>static Util::String <strong>TypeToString</strong> (Type t)</td>
<td>convert type to string</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>
## 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>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetNumArrayElements</strong> (SizeT n)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set number of array elements</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.
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.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ShaderVariableInstanceBase ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~ShaderVariableInstanceBase ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp; GetShaderVariable () const</code></td>
<td>Get the associated shader variable</td>
</tr>
<tr>
<td><code>void Apply ()</code></td>
<td>Apply local value to shader variable</td>
</tr>
<tr>
<td><code>void SetInt (int value)</code></td>
<td>Set int value</td>
</tr>
<tr>
<td><code>void SetIntArray (const int *values, SizeT count)</code></td>
<td>Set int array values</td>
</tr>
<tr>
<td><code>void SetFloat (float value)</code></td>
<td>Set float value</td>
</tr>
<tr>
<td><code>void SetFloatArray (const float *values, SizeT count)</code></td>
<td>Set float array values</td>
</tr>
<tr>
<td><code>void SetVector (const Math::float4 &amp;value)</code></td>
<td>Set float4 value</td>
</tr>
<tr>
<td><code>void SetVectorArray (const Math::float4 *values, SizeT count)</code></td>
<td>Set float4 array values</td>
</tr>
<tr>
<td><code>void SetMatrix (const Math::matrix44 &amp;value)</code></td>
<td>Set matrix44 value</td>
</tr>
<tr>
<td><code>void SetMatrixArray (const Math::matrix44 *values, SizeT count)</code></td>
<td>Set matrix44 array values</td>
</tr>
<tr>
<td><code>void SetBool (bool value)</code></td>
<td>Set bool value</td>
</tr>
<tr>
<td><code>void SetBoolArray (const bool *values,</code></td>
<td>Set bool array values</td>
</tr>
</tbody>
</table>
void 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 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

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.

```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::ShaderVariationBase
Base::ShaderVariationBase Class Reference

#include <shadervariationbase.h>

Inheritance diagram for Base::ShaderVariationBase:
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
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ShaderVariationBase ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>virtual ~ShaderVariationBase ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>const Name &amp; GetName () const</code></td>
<td>get the shader variation's name</td>
</tr>
<tr>
<td><code>CoreGraphics::ShaderFeature::Mask GetFeatureMask () const</code></td>
<td>get the feature bit mask of this variation</td>
</tr>
<tr>
<td><code>SizeT GetNumPasses () const</code></td>
<td>get number of passes in this variation</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>IsA (const Util::FourCC &amp;rttiFourCC)</code></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>bool</td>
<td>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>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>SetName (const Name &amp;n)</code></td>
<td><code>set variation name</code></td>
</tr>
<tr>
<td><code>SetFeatureMask (CoreGraphics::ShaderFeature::Mask m)</code></td>
<td><code>set feature bit mask of this variation</code></td>
</tr>
<tr>
<td><code>SetNumPasses (SizeT n)</code></td>
<td><code>set number of passes</code></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 <shaperendererbase.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 Types

<table>
<thead>
<tr>
<th>enum</th>
<th>ShapeType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shape type</td>
</tr>
</tbody>
</table>
### 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><strong>virtual ~ShapeRendererBase ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Open ()</strong></td>
<td>open the shape renderer</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close the shape renderer</td>
</tr>
<tr>
<td><strong>bool IsOpen () const</strong></td>
<td>return true if open</td>
</tr>
<tr>
<td><strong>void Begin ()</strong></td>
<td>begin drawing shapes</td>
</tr>
<tr>
<td><strong>void DrawShape (const Math::matrix44 &amp;modelTransform, ShapeType shapeType, const Math::float4 &amp;color)</strong></td>
<td>draw a unit box</td>
</tr>
<tr>
<td><strong>void DrawPrimitives (const Math::matrix44 &amp;modelTransform,</strong> CoreGraphics::PrimitiveTopology::Code topology,** SizeT numPrimitives,** float <em>vertices,</em>* SizeT vertexWidth,** const Math::float4 &amp;color)**</td>
<td>draw primitives</td>
</tr>
<tr>
<td><strong>void DrawIndexedPrimitives (const Math::matrix44 &amp;modelTransform,</strong> CoreGraphics::PrimitiveTopology::Code topology,** SizeT numPrimitives,** float <em>vertices,</em>* SizeT numVertices,** SizeT vertexWidth,** void <em>indices,</em>* CoreGraphics::IndexType::Code indexType,** const Math::float4 &amp;color)**</td>
<td>draw indexed primitives</td>
</tr>
<tr>
<td><strong>void End ()</strong></td>
<td>end drawing shapes</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</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>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(Rtti &amp;rtti) const</code></td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(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(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(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(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(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>
<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
This method should be called as the very last before an application exits.
Base::StreamTextureSaverBase
#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><strong>SetStream</strong> (const Ptr&lt; IO::Stream &gt; &amp;stream)</td>
<td>set stream to save to</td>
</tr>
<tr>
<td><strong>GetStream</strong> () const</td>
<td>get save-stream</td>
</tr>
<tr>
<td><strong>SetFormat</strong> (CoreGraphics::ImageFileFormat::Code fmt)</td>
<td>set file format (default is JPG)</td>
</tr>
<tr>
<td>CoreGraphics::ImageFileFormat::Code <strong>GetFormat</strong> () const</td>
<td>get file format</td>
</tr>
<tr>
<td><strong>SetMipLevel</strong> (IndexT mipLevel)</td>
<td>set the mip level to save (default is 0, the top level)</td>
</tr>
<tr>
<td>IndexT <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>
</tbody>
</table>
```plaintext
| 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!)
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::TextureBase
#include <texturebase.h>

Inheritance diagram for Base::TextureBase:

```
Core::RefCounted

Resources::Resource

Base::ResourceBase

Base::TextureBase

Direct3D9::D3DTexture

CoreGraphics::Texture
```
Detailed Description

The base class for texture objects.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>Enum</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</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 <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>TextureBase ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~TextureBase ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>GetType () const</code></td>
<td>get texture type</td>
</tr>
<tr>
<td><code>GetWidth () const</code></td>
<td>get width of texture</td>
</tr>
<tr>
<td><code>GetHeight () const</code></td>
<td>get height of texture (if 2d or 3d texture)</td>
</tr>
<tr>
<td><code>GetDepth () const</code></td>
<td>get depth of texture (if 3d texture)</td>
</tr>
<tr>
<td><code>GetNumMipLevels () const</code></td>
<td>get number of mip levels</td>
</tr>
<tr>
<td><code>GetPixelFormat () const</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>Unmap (IndexT mipLevel)</code></td>
<td>unmap texture after 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>UnmapCubeFace (CubeFace face, IndexT mipLevel)</code></td>
<td>unmap cube map face after CPU access</td>
</tr>
<tr>
<td><code>GetUsage () const</code></td>
<td>get resource usage type</td>
</tr>
<tr>
<td><code>GetAccess () const</code></td>
<td>get cpu access type</td>
</tr>
<tr>
<td>void <code>SetAsyncEnabled (bool b)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsAsyncEnabled () const</td>
<td>Return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td>SetResourceId (const ResourceId &amp;id)</td>
<td>Set the resource identifier</td>
</tr>
<tr>
<td>GetResourceId () const</td>
<td>Get the resource identifier</td>
</tr>
<tr>
<td>SetLoader (const Ptr&lt;ResourceLoader&gt; &amp;loader)</td>
<td>Set optional resource loader</td>
</tr>
<tr>
<td>GetLoader () const</td>
<td>Get optional resource loader</td>
</tr>
<tr>
<td>SetSaver (const Ptr&lt;ResourceSaver&gt; &amp;saver)</td>
<td>Set optional resource saver</td>
</tr>
<tr>
<td>GetSaver () const</td>
<td>Get optional resource saver</td>
</tr>
<tr>
<td>GetUseCount () const</td>
<td>Get current use count</td>
</tr>
<tr>
<td>Load ()</td>
<td>Load the resource</td>
</tr>
<tr>
<td>Unload ()</td>
<td>Unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td>GetState () const</td>
<td>Get current state</td>
</tr>
<tr>
<td>IsLoaded () const</td>
<td>Return true if current state is Loaded</td>
</tr>
<tr>
<td>IsPending () const</td>
<td>Return true if current state is Pending</td>
</tr>
<tr>
<td>LoadFailed () const</td>
<td>Return true if current state is Failed</td>
</tr>
<tr>
<td>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>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</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

void SetType (Type t)  
set texture type

void SetWidth (SizeT w)  
set texture width

void SetHeight (SizeT h)  
set texture height

void SetDepth (SizeT d)  
set texture depth

void SetNumMipLevels (SizeT n)  
set number of mip levels

void SetPixelFormat (CoreGraphics::PixelFormat::Code f)  
set pixel format

void SetUsage (Usage usage)  
set resource usage type

void SetAccess (Access access)  
set resource cpu access type

void SetState (State s)  
set current state

void IncrUseCount ()  
increment use count

void DecrUseCount ()  
decrement use count
## Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>MapInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>access info filled by Map methods</em> 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.

```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 `Base::MeshBase`, `Base::VertexBufferBase`, `CoreGraphics::CPUIndexBuffer`, `CoreGraphics::CPUVertexBuffer`, `Direct3D9::D3D9IndexBuffer`, `Direct3D9::D3D9Shader`, `Direct3D9::D3D9Texture`, `Direct3D9::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.
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

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:

```
    Core::RefCounted
        
    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>virtual bool Open ()</td>
<td>Open the transform device</td>
</tr>
<tr>
<td>virtual 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>virtual void ApplyViewSettings ()</td>
<td>Apply view dependent settings</td>
</tr>
<tr>
<td>virtual void ApplyModelTransforms ()</td>
<td>Apply any model transform needed, implementation is platform dependend</td>
</tr>
<tr>
<td>virtual 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>Get current model transform</td>
</tr>
</tbody>
</table>
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

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!)
```
### 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>
</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>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>VertexBufferBase()</code></td>
<td>constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>~VertexBufferBase()</code></td>
<td>destructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Unload()</code></td>
<td>unload the resource, or cancel the pending load</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Map(MapType mapType)</code></td>
<td>map the vertices for CPU access</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Unmap()</code></td>
<td>unmap the resource</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetVertexLayout()</code></td>
<td>get the vertex layout</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetNumVertices()</code></td>
<td>get number of vertices in the buffer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetUsage()</code></td>
<td>get resource usage type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetAccess()</code></td>
<td>get cpu access type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetAsyncEnabled(bool b)</code></td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsAsyncEnabled()</code></td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetResourceId(const ResourceId &amp;id)</code></td>
<td>set the resource identifier</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetResourceId()</code></td>
<td>get the resource identifier</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetLoader(const Ptr&lt;ResourceLoader&gt; &amp;loader)</code></td>
<td>set optional resource loader</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetLoader()</code></td>
<td>get the resource identifier</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>get optional resource loader</code></td>
<td></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>State GetState () const</code></td>
<td>get current state</td>
</tr>
<tr>
<td><code>bool IsLoaded () const</code></td>
<td>return true if current state is Loaded</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><strong>bool</strong></td>
<td><strong>IsA (const Rtti &amp;rtti) const</strong></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>
<tr>
<td><strong>bool</strong></td>
<td><strong>IsA (const Util::String &amp;rttiName) const</strong></td>
</tr>
<tr>
<td></td>
<td><code>return true if this object is instance of given class, or a derived class, by string</code></td>
</tr>
<tr>
<td><strong>bool</strong></td>
<td><strong>IsA (const Util::FourCC &amp;rttiFourCC) const</strong></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><strong>const Util::String &amp;</strong></td>
<td><strong>GetClassName () const</strong></td>
</tr>
<tr>
<td></td>
<td><code>get the class name</code></td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
<td><strong>GetClassFourCC () const</strong></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>DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dump refcounting leaks, call at end of application (NEBUlus3_DBG buiIlS onLy!)</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetVertexLayout</code></td>
<td>(const Ptr&lt; CoreGraphics::VertexLayout &gt; &amp;vertexLayout) set vertex layout (set by resource loader)</td>
</tr>
<tr>
<td><code>SetNumVertices</code></td>
<td>(SizeT numVertices) set number of vertices (set by resource loader)</td>
</tr>
<tr>
<td><code>SetUsage</code></td>
<td>(Usage usage) set resource usage type</td>
</tr>
<tr>
<td><code>SetAccess</code></td>
<td>(Access access) set resource cpu access type</td>
</tr>
<tr>
<td><code>SetState</code></td>
<td>(State s) set current state</td>
</tr>
<tr>
<td><code>IncrUseCount</code></td>
<td>increment use count</td>
</tr>
<tr>
<td><code>DecrUseCount</code></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

```c
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 CoreGraphics::CPUVertexBuffer, and Direct3D9::D3D9VertexBuffer.

```c
void Base::*VertexBufferBase::*Unmap ( )
```

unmap the resource

Give up CPU access on the vertex buffer content.

Reimplemented in CoreGraphics::CPUVertexBuffer, and Direct3D9::D3D9VertexBuffer.

```c
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.

```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
Base::VertexLayoutBase Class Reference

#include <vertexlayoutbase.h>

Inheritance diagram for Base::VertexLayoutBase:
Detailed Description

**Base** class for VertexLayout. A VertexLayout object describes what components a vertex has and offers some utility functions. Vertex layout objects are generally shared and created through the VertexLayoutServer (this is more efficient on some platforms).

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VertexLayoutBase ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~VertexLayoutBase ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>Setup (const Util::Array<a href="">CoreGraphics::VertexComponent</a> &amp;c)</strong></td>
<td>Setup the vertex layout</td>
</tr>
<tr>
<td><strong>IsValid () const</strong></td>
<td>Return true if valid has been setup</td>
</tr>
<tr>
<td><strong>GetNumComponents () const</strong></td>
<td>Get number of components</td>
</tr>
<tr>
<td><strong>GetComponentAt (IndexT i) const</strong></td>
<td>Get vertex component at index</td>
</tr>
<tr>
<td><strong>HasComponent (CoreGraphics::VertexComponent::SemanticName, IndexT semIndex) const</strong></td>
<td>Return true if vertex component exists</td>
</tr>
<tr>
<td><strong>FindComponent (CoreGraphics::VertexComponent::SemanticName, IndexT semIndex) const</strong></td>
<td>Get index of vertex component by semantics</td>
</tr>
<tr>
<td><strong>GetVertexByteSize () const</strong></td>
<td>Get the vertex stride in number of bytes</td>
</tr>
<tr>
<td><strong>GetComponentByteOffset (CoreGraphics::VertexComponent::SemanticName, IndexT semIndex) const</strong></td>
<td>Get component offset from start of vertex</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>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>return true if this object is instance of given class</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool 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>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool 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>
</tr>
</tbody>
</table>

<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) const</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;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>
</tbody>
</table>
Static Public Member Functions

```c
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>Discard</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>discard the vertex layout object</em></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>
</tr>
</thead>
<tbody>
<tr>
<td>get sharing signature for a set of vertex components</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

IndexT
Base::VertexLayoutBase::GetComponentByteOffset (CoreGraphics::VertexComponent::SemanticName IndexT
)

get component offset from start of vertex

Returns the component offset in bytes from the beginning of a vertex to the start of the given vertex components. Returns InvalidIndex if the vertex component doesn't exist!

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.

dump 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
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>CategoryManager ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~CategoryManager ()</td>
</tr>
<tr>
<td>destructor</td>
</tr>
<tr>
<td>virtual void OnActivate ()</td>
</tr>
<tr>
<td>called when attached to game server</td>
</tr>
<tr>
<td>virtual void OnDeactivate ()</td>
</tr>
<tr>
<td>called when removed from game server</td>
</tr>
<tr>
<td>void CommitChangesToDatabase ()</td>
</tr>
<tr>
<td>commit changes back into database</td>
</tr>
<tr>
<td>void LoadInstances (const Util::String &amp;levelName)</td>
</tr>
<tr>
<td>load all instances with the given level attribute</td>
</tr>
<tr>
<td>Util::Array&lt; Ptr &lt; Db::Dataset &gt; &gt; FindInstances (const Util::String &amp;levelName)</td>
</tr>
<tr>
<td>find all instances with the given level attribute</td>
</tr>
<tr>
<td>Entry CreateDummyInstance ()</td>
</tr>
<tr>
<td>create a dummy instance which will never be saved to the database</td>
</tr>
<tr>
<td>Entry CreateInstanceFromTemplate (const Util::String &amp;categoryName, const Util::String &amp;id)</td>
</tr>
<tr>
<td>create a new instance from a template</td>
</tr>
<tr>
<td>Entry CreateInstanceFromTemplateAsCategory (const Util::String &amp;categoryName, const Util::String &amp;id, const Util::String &amp;targetCategory)</td>
</tr>
<tr>
<td>create a new instance from a template as a different category</td>
</tr>
<tr>
<td>Entry CreateInstanceFromAttrs (const Util::String &amp;categoryName, const Util::Array &lt; Attr::Attribute &gt; &amp;attrs)</td>
</tr>
<tr>
<td>create a new instance from a template and override attributes</td>
</tr>
<tr>
<td>Entry CreateInstanceFromInstance (const Entry &amp;source)</td>
</tr>
<tr>
<td>create a new instance as a copy from another instance of the same category</td>
</tr>
<tr>
<td>Entry</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td><strong>Entry</strong></td>
</tr>
<tr>
<td><code>Util::Array&lt; Entry &gt;</code></td>
</tr>
<tr>
<td><code>Util::Array&lt; Entry &gt;</code></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td><code>Ptr&lt; Game::Entity &gt;</code></td>
</tr>
<tr>
<td>bool</td>
</tr>
<tr>
<td><code>SizeT</code></td>
</tr>
<tr>
<td>const <code>Category</code> &amp;</td>
</tr>
<tr>
<td>const <code>Category</code> &amp;</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
</tbody>
</table>
add a category attribute

```cpp
void EndAddCategoryAttrs ()
end adding category attributes
```

```cpp
bool HasTemplateTable (const Util::String &categoryName) const
return true if a template table for a category exists
```

```cpp
const Ptr < Db::ValueTable > & GetTemplateTable (const Util::String &categoryName) const
get template value table by category name
```

```cpp
Entry FindTemplate (const Util::String &categoryName, const Util::String &id) const
find a template entry by category name and id
```

```cpp
Entry FindTemplateByAttr (const Attr::Attribute &attr, const Util::String &categoryName="") const
find a single template by attribute
```

```cpp
bool HasInstanceTable (const Util::String &categoryName) const
return true if an instance table for a category exists
```

```cpp
const Ptr < Db::ValueTable > & GetInstanceTable (const Util::String &categoryName) const
get instance value table by category name
```

```cpp
void DuplicateLevel (const Util::String &levelName, const Util::String &duplicateName)
duplicates a complete level in the database
```

```cpp
void DeleteLevel (const Util::String &levelName)
deletes a complete level from database
```

```cpp
void RenameLevel (const Util::String &levelName, const Util::String &newLevelName)
renames a complete level in the database
```

```cpp
bool LevelExists (const Util::String &levelName)
return true if level is found in _Instance_Levels table
```

```cpp
bool IsActive () const
return true if currently active
```

```cpp
virtual void OnBeginFrame ()
called before frame by the game server
```

```cpp
virtual void OnFrame ()
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void OnEndFrame ()</td>
<td>called per-frame by the game server</td>
</tr>
<tr>
<td>virtual void OnLoad ()</td>
<td>called after frame by the game server</td>
</tr>
<tr>
<td>virtual void OnSave ()</td>
<td>called after loading game state</td>
</tr>
<tr>
<td>virtual void OnStart ()</td>
<td>called before saving game state</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>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>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>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 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
```
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td><em>describes a category</em> <a href="#">More...</a></td>
</tr>
<tr>
<td>Entry</td>
<td><em>describes an entry in a value table</em> <a href="#">More...</a></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).
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 from a template and override attributes

This creates a new instance from scratch and fills it with the provided attributes.
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.

```
CategoryManager::Entry
BaseGameFeature::CategoryManager::CreateInstanceFromInstanceAsCategory ( const 
  Entry & source
  const Util::String & targetCategory
  bool createMissingAttributes = false
)
```

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.

```
Util::Array< CategoryManager::Entry >
BaseGameFeature::CategoryManager::GetInstancesByAttr ( const Attr::Attribute & attrs,
  bool loadedOnly,
  bool firstMatchOnly
)
```

find and/or create new instance by matching a key attribute, this method can be quite slow!

Create new instances by a matching attribute. This method can do several database access, and thus can be slow if the requested attribute doesn't map to an indexed database column.

If you know that there is only one instance in the database, set the firstMatchOnly parameter to true to avoid unecessary database accesses.

Matching instances which already loaded will not be correctly ignored, this is done by checking against the primary attribute (Attr::Guid) of the table.
Util::Array< CategoryManager::Entry >

BaseGameFeature::CategoryManager::GetInstancesByAttrs(
    const Util::Array< Attr::Attribute >& attrs,
    bool loadedOnly,
    bool firstMatchOnly
);

find and/or create new instance by matching a key attribute, this method can be quite slow!

Create new instances by a multiple matching attributes.

void
BaseGameFeature::CategoryManager::DeleteInstance( const Entry& entry )

delete an instance identified by GUID

This deletes an instance from the database.

int
BaseGameFeature::CategoryManager::GetNumInstances() const

get current overall number of instances

This returns the overall number of loaded instances.

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.
get instance entity pointer (can be 0!)

Get the game entity pointer associated with an instance.

```cpp
void BaseGameFeature::CategoryManager::BeginAddCategoryAttrs(const Util::String categoryName )
```

begin adding category attributes

Begin adding category attributes.

```cpp
void BaseGameFeature::CategoryManager::AddCategoryAttr(const Attr::AttrId attrId )
```

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.

```cpp
void BaseGameFeature::CategoryManager::EndAddCategoryAttrs()
```

derend adding category attributes

End adding attributes to a category.

```cpp
CategoryManager::Entry
BaseGameFeature::CategoryManager::FindTemplateByAttr(const Attr::Attribute attr, const Util::String & categoryName = ""
)
```

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.

```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<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**.

```cpp
void Messaging::Dispatcher::AttachPort(const Ptr<const 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

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.
dump 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

#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>const Util::String &amp; <strong>GetName () const</strong></td>
<td>get category name</td>
</tr>
<tr>
<td>bool <strong>IsVirtual () const</strong></td>
<td>return true if virtual category (starting with a . in the db.xml)</td>
</tr>
<tr>
<td>bool <strong>IsSpecial () const</strong></td>
<td>return true if this is a special category (e.g. _Environment)</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetTemplateTableName () const</strong></td>
<td>get template table name</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>GetInstanceTableName () const</strong></td>
<td>get instance table name</td>
</tr>
<tr>
<td>bool <strong>HasTemplateDataset () const</strong></td>
<td>return true if the category has a template dataset</td>
</tr>
<tr>
<td>bool <strong>HasInstanceDataset () const</strong></td>
<td>return true if the category has an instance dataset</td>
</tr>
<tr>
<td>const Ptr &lt; Db::Dataset &gt; &amp; <strong>GetTemplateDataset () const</strong></td>
<td>get pointer to template dataset</td>
</tr>
<tr>
<td>const Ptr &lt; Db::Dataset &gt; &amp; <strong>GetInstanceDataset () const</strong></td>
<td>get pointer to instance dataset</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:43 2008
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>default constructor</td>
</tr>
<tr>
<td><strong>Entry (const Util::String &amp;categoryName, const Ptr<a href="">Db::ValueTable</a> &amp;valueTable, IndexT rowIndex)</strong></td>
<td>constructor with value table and row index</td>
</tr>
<tr>
<td>bool IsValid () const</td>
<td>return true if valid</td>
</tr>
<tr>
<td>const Util::String &amp; Category () const</td>
<td>get category name</td>
</tr>
<tr>
<td>const Ptr<a href="">Db::ValueTable</a> &amp; Values () const</td>
<td>get pointer to value table</td>
</tr>
<tr>
<td>IndexT RowIndex () const</td>
<td>get row index in value table</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](http://www.doxygen.org) at Tue Feb 19 12:16:43 2008
BaseGameFeature::EntityLoader
BaseGameFeature::EntityLoader Class Reference

#include <entityloader.h>

Inheritance diagram for 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><strong>virtual bool</strong></td>
<td><strong>Load</strong> (const <strong>Util::Array</strong>&lt; <strong>Util::String</strong> &gt; &amp;activeLayers)</td>
</tr>
<tr>
<td></td>
<td>load entity objects into the level</td>
</tr>
<tr>
<td><strong>int</strong></td>
<td><strong>GetRefCount</strong> () const</td>
</tr>
<tr>
<td></td>
<td>get the current refcount</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>AddRef</strong> ()</td>
</tr>
<tr>
<td></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>Release</strong> ()</td>
</tr>
<tr>
<td></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><strong>bool</strong></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><strong>bool</strong></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><strong>bool</strong></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><strong>bool</strong></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><strong>bool</strong></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,</td>
</tr>
<tr>
<td></td>
<td>by string</td>
</tr>
<tr>
<td><strong>bool</strong></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,</td>
</tr>
<tr>
<td></td>
<td>by fourcc</td>
</tr>
<tr>
<td>**const <strong>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><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>bool</th>
<th><strong>EntityIsInActiveLayer</strong> (const Ptr&lt; Db::ValueTable &gt; &amp;values, IndexT rowIndex, const Util::Array&lt; Util::String &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::EntityIsActiveLayer(
    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.

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::EntityLoaderBase
#include <entityloaderbase.h>

Inheritance diagram for BaseGameFeature::EntityLoaderBase:
Detailed Description

Abstract loader helper for game entities.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EntityLoaderBase ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~EntityLoaderBase ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual bool Load (const Util::Array&lt;Util::String&gt; &amp;activeLayers)</code></td>
<td>Load entity objects into the level</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>
<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><code>EntityIsInActiveLayer</code> (const Ptr&lt; Db::ValueTable &gt; &amp;values, IndexT rowIndex, const Util::Array&lt; Util::String &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, 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.

```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 Tue Feb 19 12:16:43 2008
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><strong>EntityManager()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~EntityManager()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual <strong>SetMaxTriggerDistance(float d)</strong></td>
<td>TEMP HACK: set the maximum-trigger-distance (default is 100 meters).</td>
</tr>
<tr>
<td>float <strong>GetMaxTriggerDistance()</strong> const</td>
<td>TEMP HACK: get the maximum-trigger-distance.</td>
</tr>
<tr>
<td>virtual void <strong>OnDeactivate()</strong></td>
<td>called when removed from game server</td>
</tr>
<tr>
<td>void <strong>OnStart()</strong></td>
<td>called on start by game server</td>
</tr>
<tr>
<td>virtual void <strong>OnBeginFrame()</strong></td>
<td>called per-frame by game server</td>
</tr>
<tr>
<td>virtual void <strong>OnEndFrame()</strong></td>
<td>called per-frame by game server</td>
</tr>
<tr>
<td>virtual void <strong>OnSave()</strong></td>
<td>called before saving game state</td>
</tr>
<tr>
<td>void <strong>OnRenderDebug()</strong></td>
<td>called if a render debug visualization is requested</td>
</tr>
<tr>
<td>void <strong>Cleanup()</strong></td>
<td>cleanup the entity manager, removes/deactivates all entities</td>
</tr>
<tr>
<td>bool <strong>HasActiveEntities()</strong> const</td>
<td>return true if at least one active entity exists in the world</td>
</tr>
<tr>
<td><strong>GetEntities()</strong></td>
<td>get currently active entities</td>
</tr>
<tr>
<td>void <strong>AttachEntity(const Ptr<a href="">Game::Entity</a> &amp;entity)</strong></td>
<td>immediately attach an entity to the world</td>
</tr>
<tr>
<td>void <strong>RemoveEntity(const Ptr<a href="">Game::Entity</a> &amp;entity)</strong></td>
<td>remove an entity from the world, delayed until end of frame</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DeleteEntity</strong></td>
<td>delete an entity from the world (also deletes the entity from DB!), delayed until end of frame</td>
</tr>
<tr>
<td><strong>RemoveEntityImmediate</strong></td>
<td>remove an entity from the world, calling restrictions apply</td>
</tr>
<tr>
<td><strong>DeleteEntityImmediate</strong></td>
<td>delete an entity from the world (also deletes the entity from DB!), calling restrictions apply</td>
</tr>
<tr>
<td><strong>ExistsEntityByUniqueId</strong></td>
<td>return true if a entity with the given unique id exists</td>
</tr>
<tr>
<td><strong>GetEntityByUniqueId</strong></td>
<td>get the entity for the given unique id</td>
</tr>
<tr>
<td><strong>ExistsEntityByAttr</strong></td>
<td>return true if at least one entity exists with the given attribute</td>
</tr>
<tr>
<td><strong>GetEntitiesByAttr</strong></td>
<td>get the entities for the given attribute</td>
</tr>
<tr>
<td><strong>GetEntitiesByAttrs</strong></td>
<td>get a single entity by multiple matching attributes</td>
</tr>
<tr>
<td><strong>GetEntityByAttr</strong></td>
<td>get a single entity by a single attribute</td>
</tr>
<tr>
<td><strong>GetEntityByAttrs</strong></td>
<td>get a single entity by multiple matching attributes</td>
</tr>
<tr>
<td><strong>GetEntitiesInActivityBubble</strong></td>
<td>fill provided array with all entities inside the activity bubble</td>
</tr>
<tr>
<td><strong>OnActivate</strong></td>
<td>called when attached to game server</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>IsActive () const</strong></td>
<td>return true if currently active</td>
</tr>
<tr>
<td><strong>virtual void OnFrame ()</strong></td>
<td>called per-frame by the 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 HandleMessage (const Ptr&lt; Messaging::Message &gt; &amp;msg)</strong></td>
<td>handle a single message (distribute to ports which accept the message)</td>
</tr>
<tr>
<td><strong>void AttachPort (const Ptr&lt; Port &gt; &amp;port)</strong></td>
<td>attach a message port</td>
</tr>
<tr>
<td><strong>void RemovePort (const Ptr&lt; Port &gt; &amp;port)</strong></td>
<td>remove a message port</td>
</tr>
<tr>
<td><strong>bool HasPort (const Ptr&lt; Port &gt; &amp;port) const</strong></td>
<td>return true if a port exists</td>
</tr>
<tr>
<td><strong>virtual void SetupAcceptedMessages ()</strong></td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td><strong>void AttachHandler (const Ptr&lt; Handler &gt; &amp;h)</strong></td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td><strong>void RemoveHandler (const Ptr&lt; Handler &gt; &amp;h)</strong></td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td><strong>SizeT GetNumHandlers () const</strong></td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td><strong>const Ptr&lt; Handler &gt; &amp; GetHandlerAtIndex (IndexT i) const</strong></td>
<td>get a message handler by index</td>
</tr>
<tr>
<td><strong>virtual void Send (const Ptr&lt; Message &gt; &amp;msg)</strong></td>
<td>send a message to the port</td>
</tr>
<tr>
<td><strong>const Util::Array&lt; const Id * &gt; &amp; GetAcceptedMessages () const</strong></td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td><strong>bool AcceptsMessage (const Id &amp;msgId) const</strong></td>
<td>return true if port accepts this msg</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)</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) 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 Types

<table>
<thead>
<tr>
<th>enum</th>
<th>DelayedJobType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>delayed job types</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void StartEntities()</code></td>
<td>call <code>OnStart()</code> on all entities</td>
</tr>
<tr>
<td><code>void ActivateEntity(const Ptr&lt;Game::Entity&gt; &amp;entity)</code></td>
<td>activate an entity</td>
</tr>
<tr>
<td><code>void DeactivateEntity(constPtr&lt;Game::Entity&gt; &amp;entity)</code></td>
<td>deactivate an entity</td>
</tr>
<tr>
<td><code>void AddDelayedJob(DelayedJobType type, const Ptr&lt;Game::Entity&gt; &amp;entity)</code></td>
<td>add a delayed job (Remove or Delete entity)</td>
</tr>
<tr>
<td><code>void HandleDelayedJobs()</code></td>
<td>handle all queued up delayed jobs, called at beginning and end of frame</td>
</tr>
<tr>
<td><code>void RemoveNullEntriesFromArrays()</code></td>
<td>remove all null entries from the internal arrays</td>
</tr>
<tr>
<td><code>void UpdateTriggeredEntities()</code></td>
<td>update both arrays</td>
</tr>
<tr>
<td><code>bool IsInFocus(const Ptr&lt;Game::Entity&gt; &amp;entity, Math::point &amp;focusEntityPos)</code></td>
<td>check if entity is near focus entity</td>
</tr>
<tr>
<td><code>void RemoveEntityFromTriggered(const Ptr&lt;Game::Entity&gt; &amp;entity)</code></td>
<td>remove entity from triggered/untriggered arrays</td>
</tr>
<tr>
<td><code>void RegisterMessage(const Id &amp;msgId)</code></td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Data Structures

class DelayedJob

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) const

get the entity for the given unique id

Return entity by unique id or an invalid Ptr<> if the entity doesn't exist.

bool BaseGameFeature::EntityManager::ExistsEntityByAttr(const Attr::Attribute &attr) const

return true if at least one entity exists with the given attribute

Returns true if at least one entity exists with a matching attribute.

Util::Array<Ptr<Game::Entity>> BaseGameFeature::EntityManager::GetEntitiesByAttr(const Attr::Attribute &attr, bool onlyFirstEntity = false)

get the entities for the given attribute

Returns all entities which match a given attribute. If only the first entity is interesting (if you know that there will only one result) the onlyFirstEntity flag can be used to stop searching after the first match.

Util::Array<Util::Array<Ptr<Game::Entity>>> BaseGameFeature::EntityManager::GetEntitiesByAttrs(const Attr::Attribute &attr, Attr::Attribute &attr)
bool onlyFirstEntity = false)

get a single entity by multiple matching attributes

Returns all entities which match multiple attributes. If only the first entity is interesting (if you know that there will only one result) the onlyFirstEntity flag can be used to stop searching after the first match.

```cpp
void BaseGameFeature::EntityManager::GetEntitiesInActivityBubble (Util::Array<Ptr<Game::Entity>>& outEntities)
```

fill provided array with all entities inside the activity bubble

This 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

```cpp
void BaseGameFeature::EntityManager::StartEntities () [protected]
```

call **OnStart()** on all entities

Invoke the **OnStart()** method on all active entities.

```cpp
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) 
```
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()
```

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::RemoveNullEntriesFromArrays()
```

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()
```

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 shouldnt be triggered.

```cpp
bool BaseGameFeature::EntityManager::IsInFocus(const Ptr<Game::Entity> &curEntity, Math::point &focusEntityPos)
```
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)
```

remove entity from triggerd/untriggered arrays

Immediately remove an entity from the entity manager. This method is more restrictive then `RemoveEntity()` because it must not be called from inside the game loop.

```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::CategoryManager`, `BaseGameFeature::EnvQueryManager`, and `BaseGameFeature::GlobalAttrsManager`.

```cpp
void Messaging::Dispatcher::HandleMessage(const Ptr<Messaging::Message> & msg)
```

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`.

```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.
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.
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><strong>DelayedJob ()</strong></td>
<td>default constructor</td>
</tr>
<tr>
<td><strong>DelayedJob (DelayedJobType t, const Ptr&lt; Game::Entity &gt; &amp;e)</strong></td>
<td>constructor</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

BaseGameFeature::EnvEntityManager
#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><code>~EnvEntityManager()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><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><code>EnvEntityExists(const Util::String&amp;id)</code></td>
<td>const</td>
</tr>
<tr>
<td><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><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; GetGraphicsEntities(const Util::String&amp;id)</code></td>
<td>get graphics entities by id (may return empty array)</td>
</tr>
<tr>
<td><code>ClearEnvEntity()</code></td>
<td>clear environment entity</td>
</tr>
<tr>
<td><code>OnActivate()</code></td>
<td>called when attached to game server</td>
</tr>
<tr>
<td><code>IsActive()</code></td>
<td>return true if currently active</td>
</tr>
<tr>
<td><code>OnBeginFrame()</code></td>
<td>called before frame by the game server</td>
</tr>
<tr>
<td><code>OnFrame()</code></td>
<td>called per-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<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

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
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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></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 (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>bool <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>bool <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>const <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 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>RegisterMessage</strong> (const <em>Id</em> &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 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)
```

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()

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::CategoryManager, BaseGameFeature::EnvQueryManager, and BaseGameFeature::GlobalAttrsManager.

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() [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`.

```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.
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.
The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:43 2008
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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool</td>
<td><strong>Load</strong> (const Util::Array<a href="">Util::String</a> &amp;activeLayers)</td>
</tr>
<tr>
<td></td>
<td>load environment objects into the level</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, 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</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>EntityIsInActiveLayer (const Ptr&lt; Db::ValueTable &gt; &amp;values, IndexT rowIndex, const Util::Array&lt; Util::String &gt; &amp;levelActiveLayers) const</td>
<td></td>
</tr>
<tr>
<td>return true if entity at current Db::Reader pos is in active layer</td>
<td></td>
</tr>
</tbody>
</table>
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::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><code>EnvQueryManager()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~EnvQueryManager()</code></td>
<td>destructor</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>virtual void OnFrame()</code></td>
<td>called per-frame by game server</td>
</tr>
<tr>
<td><code>const Physics::FilterSet &amp; GetMouseExcludeSet()</code></td>
<td>get a const Ref on the MouseExcludeSet</td>
</tr>
<tr>
<td><code>void SetMouseExcludeSet(const Physics::FilterSet &amp;mouseExcludeSet)</code></td>
<td>set a new MouseExcludeSet</td>
</tr>
<tr>
<td><code>Game::Entity * GetEntityUnderMouse()</code></td>
<td>get the entity under the mouse cursor</td>
</tr>
<tr>
<td><code>Physics::MaterialType GetMaterialUnderMouse()</code></td>
<td>get the material under the mouse cursor</td>
</tr>
<tr>
<td><code>const Math::point &amp; GetMousePos3d()</code></td>
<td>get the mouse position in the 3d world</td>
</tr>
<tr>
<td><code>Math::line ComputeMouseWorldRay(const Math::float2 &amp;mousePos, float length, const Ptr&lt;Graphics::View&gt; &amp;view)</code></td>
<td>compute 3d ray thru mouse pos</td>
</tr>
<tr>
<td><code>const Math::vector &amp; GetUpVector()</code></td>
<td>get the upVector of the face under the mouse cursor</td>
</tr>
<tr>
<td><code>bool HasMouseIntersection()</code></td>
<td>return true if mouse is over &quot;something&quot;</td>
</tr>
<tr>
<td><code>Util::Array&lt;Ptr&lt;Game::Entity&gt;&gt; GetEntitiesInSphere(const Math::point &amp;midPoint, float radius)</code></td>
<td>get all entities in a given spherical area</td>
</tr>
<tr>
<td><code>Util::Array&lt;Ptr&lt;Game::Entity&gt;&gt; GetEntitiesInBox(const Math::vector &amp;scale,</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>const Math::matrix44 &amp;m)</strong></td>
<td>get all entities in a given box shaped area</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 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&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>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></td>
</tr>
<tr>
<td>Function/Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>send a message to the port</td>
<td></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)</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, 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>
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
const Physics::FilterSet & BaseGameFeature::EnvQueryManager::GetMouseExcludeSet() const
get a const Ref on the MouseExcludeSet
```
```cpp
void BaseGameFeature::EnvQueryManager::SetMouseExcludeSet(const Physics::FilterSet & mouseExcludeSet)
set a new MouseExcludeSet
```
```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
Physics::MaterialType BaseGameFeature::EnvQueryManager::GetMaterialUnderMouse() const
get the material under the mouse cursor
This returns the MaterialType of the entity under the mouse, or
InvalidMaterial 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 mouse cursor. If the mouse doesn't intersect, the result will be undefined, and the method

`HasMouseIntersection()` returns false.

```cpp
bool
BaseGameFeature::EnvQueryManager::HasMouseIntersection() const
```

return 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, )
```
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() [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.
```
Messaging::Dispatcher::AttachPort (Port port) [inherited]

attach a message port

Attach a new message port.

Parameters:

- **port** pointer to a message port object

void Messaging::Dispatcher::RemovePort (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 (Port & port) const [inherited]

return true if a port exists

Return true if a port is already attached.

void Messaging::Port::AttachHandler (Handler & h) [inherited]

attach a message handler to the port

Attach a message handler to the port.

void 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.

**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::FactoryManager
BaseGameFeature::FactoryManager Class Reference

#include <factorymanager.h>

Inheritance diagram for BaseGameFeature::FactoryManager:
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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong><code>FactoryManager ()</code></strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~FactoryManager ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual Ptr <code>&lt;Game::Entity&gt;</code> CreateEntityByClassName (const Util::String &amp;cppClassName) 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 Ptr <code>&lt;Game::Entity&gt;</code> CreateEntityByCategory (const Util::String &amp;categoryName, const Ptr<a href="">Db::ValueTable</a> &amp;attrTable, IndexT attrTableRowIndex) const</td>
<td>create a new entity from its category name</td>
</tr>
<tr>
<td>virtual Ptr <code>&lt;Game::Entity&gt;</code> CreateEntityByAttrs (const Util::String &amp;categoryName, const Util::Array<a href="">Attr::Attribute</a> &amp;attrs) const</td>
<td>create a new entity from scratch and initialize it with the provided attributes</td>
</tr>
<tr>
<td>virtual Ptr <code>&lt;Game::Entity&gt;</code> CreateEntityByTemplate (const Util::String &amp;categoryName, const Util::String &amp;templateName) const</td>
<td>create a new entity from a database template entry</td>
</tr>
<tr>
<td>virtual Ptr <code>&lt;Game::Entity&gt;</code> CreateEntityByTemplateAsCategory (const Util::String &amp;categoryName, const Util::String &amp;templateName, const Util::String &amp;targetCategory) const</td>
<td>create a new entity from a database template entry, and add it into a different category</td>
</tr>
<tr>
<td>virtual Ptr <code>&lt;Game::Entity&gt;</code> CreateEntityByEntity (const Ptr<a href="">Game::Entity</a> &amp;sourceEntity) const</td>
<td>create a new entity cloning an existing one</td>
</tr>
<tr>
<td>virtual Ptr <code>&lt;Game::Entity&gt;</code> CreateEntityByEntityAsCategory (const Ptr<a href="">Game::Entity</a> &amp;sourceEntity, const Util::String &amp;targetCategory, bool createMissingAttributes=false) const</td>
<td>create a new entity in a different category cloning an existing one</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>[virtual Ptr &lt; Game::Entity &gt; CreateEntityByKeyAttr(const Attr::Attribute &amp;key) const]</td>
<td>create new entity from world database using any key attribute</td>
</tr>
<tr>
<td>[virtual Ptr &lt; Game::Entity &gt; CreateEntityByGuid(const Util::Guid &amp;guid) const]</td>
<td>create new entity from world database using GUID as key attribute</td>
</tr>
<tr>
<td>[virtual Ptr &lt; Game::Property &gt; CreateProperty(const Util::String &amp;type) const]</td>
<td>create a new property by type name, extend in subclass for new types</td>
</tr>
<tr>
<td>[void AddProperties(const Ptr<a href="">Game::Entity</a> &amp;entity, const Util::String &amp;categoryName) const]</td>
<td>add properties to entity according to blue print</td>
</tr>
<tr>
<td>[virtual void OnActivate()]</td>
<td>called when attached to game server</td>
</tr>
<tr>
<td>[virtual void OnDeactivate()]</td>
<td>called when removed from game server</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>
</tbody>
</table>
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

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)
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>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>bool</th>
<th>ParseBluePrints ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>parse entity blueprints file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IndexT</th>
<th>FindBluePrint (const Util::String &amp;entityType) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>find blueprint index by property type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>RegisterMessage (const Id &amp;msgId)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>register a single accepted message</td>
</tr>
</tbody>
</table>
Data Structures

struct **BluePrint**

*an entity blueprint, these are created by ParseBlueprints() More...*
Member Function Documentation

Ptr< Entity >
BaseGameFeature::FactoryManager::CreateEntityByClassName (const Util::String & 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 Game::Entity subclasses.

Ptr< Entity >
BaseGameFeature::FactoryManager::CreateEntityByCategory (const Util::String & categoryName, const Ptr< Db::ValueTable > & attrTable, IndexT attrTableRowIndex )

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.

Ptr< Entity >
BaseGameFeature::FactoryManager::CreateEntityByAttrs (const Util::String & categoryName, const Util::Array< Attr::Attribute > & attrs )

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.

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.

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.

create a new entity cloning an existing one

Create a entity as a clone of a existing one. A new GUID will be
Ptr&lt; Entity &gt;
BaseGameFeature::FactoryManager::CreateEntityByEntityAsCategory
(const Ptr&lt; Game::Entity &gt; & sourceEntity
const Util::String & targetCategory
bool createMissingAttributes = false)

create a new entity in a different category cloning an existing one

Create a entity as a clone of an existing one but in a different category as the original entity. A new GUID will be assigned.

Ptr&lt; Entity &gt;
BaseGameFeature::FactoryManager::CreateEntityByKeyAttr
(const Attr::Attribute & key ) const [virtual]

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.

Ptr&lt; Entity &gt;
BaseGameFeature::FactoryManager::CreateEntityByGuid
(const 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 \texttt{CreateEntityByKeyAttr()}. 

\begin{verbatim}
Ptr< Property > BaseGameFeature::FactoryManager::CreateProperty( const Util::String type ) const [virtual]
\end{verbatim}

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.

\begin{verbatim}
void BaseGameFeature::FactoryManager::AddProperties( const Ptr<Game::Entity>& entity, const Util::String& categoryName ) const
\end{verbatim}

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.

\begin{verbatim}
bool BaseGameFeature::FactoryManager::ParseBluePrints() [protected]
\end{verbatim}

parse entity blueprints file

This method parses the file data:tables/blueprints.xml into the bluePrints array.

\begin{verbatim}
IndexT BaseGameFeature::FactoryManager::FindBluePrint( const Util::String entityType ) const [protected]
\end{verbatim}

find blueprint index by property type

This method finds a blueprint index by entity type name. Returns
InvalidIndex if blueprint doesn't exist.

```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::CategoryManager`, `BaseGameFeature::EnvQueryManager`, and `BaseGameFeature::GlobalAttrsManager`.

```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::CategoryManager`, `BaseGameFeature::EntityManager`, `BaseGameFeature::EnvEntityManager`, and `BaseGameFeature::EnvQueryManager`.

```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**.

```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
Messaging::Dispatcher::HasPort (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.
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 Tue Feb
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

**BaseGameFeature::FactoryManager::BluePrint**
BaseGameFeature::FactoryManager::Blue
Struct Reference

#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:

```
+------------------+
| Core::RefCounted |
|                  |
| v
+------------------+
| Messaging::Port  |
|                  |
| v
+------------------+
| Messaging::Dispatcher |
|                     |
| v
+------------------+
| Game::Manager    |
|                  |
| v
+------------------+
| 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><code>FocusManager()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~FocusManager()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>OnFrame()</code></td>
<td>Trigger the focus manager once a frame, actual focus switches will happen here</td>
</tr>
<tr>
<td><code>SetFocusEntity(const Ptr&lt;Game::Entity&gt; &amp;entity)</code></td>
<td>Set input and camera focus to entity, null ptr allowed</td>
</tr>
<tr>
<td><code>GetFocusEntity()</code> const</td>
<td>Get common focus entity, will fail if input != camera focus entity</td>
</tr>
<tr>
<td><code>SetFocusToNextEntity()</code></td>
<td>Switch input and camera focus to next entity</td>
</tr>
<tr>
<td><code>SetInputFocusEntity(const Ptr&lt;Game::Entity&gt; &amp;entity)</code></td>
<td>Set the current input focus entity, null ptr allowed</td>
</tr>
<tr>
<td><code>GetInputFocusEntity()</code> const</td>
<td>Get the current input focus entity</td>
</tr>
<tr>
<td><code>SetInputFocusToNextEntity()</code></td>
<td>Set input focus to next entity</td>
</tr>
<tr>
<td><code>SetCameraFocusEntity(const Ptr&lt;Game::Entity&gt; &amp;entity)</code></td>
<td>Set the current camera focus entity, null ptr allowed</td>
</tr>
<tr>
<td><code>GetCameraFocusEntity()</code> const</td>
<td>Get the current camera focus entity</td>
</tr>
<tr>
<td><code>OnActivate()</code></td>
<td>Called when attached to game server</td>
</tr>
<tr>
<td><code>OnDeactivate()</code></td>
<td>Called when removed from game server</td>
</tr>
</tbody>
</table>
bool IsActive() const
return true if currently active

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

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 GetAcceptedMessages() const
null
**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>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
<td><strong>SwitchFocusEntities</strong> ()</td>
<td>actually switch focus entities</td>
</tr>
<tr>
<td><code>void</code></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 BaseGameFeature::FocusManager::OnFrame() [virtual]
```

trigger the focus manager once a frame, actual focus switches will happen here

This method is called once per frame by the game server and actually handles focus entity switches.

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::CategoryManager, BaseGameFeature::EnvQueryManager, and BaseGameFeature::GlobalAttrsManager.

```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::CategoryManager`, `BaseGameFeature::EntityManager`, `BaseGameFeature::EnvEntityManager`, and `BaseGameFeature::EnvQueryManager`.

```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 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`.

void Messaging::Dispatcher::AttachPort(const Messaging::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.
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::GlobalAttrsManager
#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><code>GlobalAttrsManager ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~GlobalAttrsManager ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual void OnActivate ()</code></td>
<td>Called when attached to game server</td>
</tr>
<tr>
<td><code>void SaveAttributes ()</code></td>
<td>Explicitly save attributes to database (not through OnSave!)</td>
</tr>
<tr>
<td><code>void LoadAttributes ()</code></td>
<td>Explicitly load attributes from database (not through OnLoad())</td>
</tr>
<tr>
<td><code>bool HasAttr (const Attr::AttrId &amp;attrId) const</code></td>
<td>Return true if global attribute exists</td>
</tr>
<tr>
<td><code>void SetString (const Attr::StringAttrId &amp;attrId, const Util::String &amp;value)</code></td>
<td>Set a global string attribute</td>
</tr>
<tr>
<td><code>const Util::String &amp; GetString (const Attr::StringAttrId &amp;attrId) const</code></td>
<td>Get a global string attribute</td>
</tr>
<tr>
<td><code>void SetInt (const Attr::IntAttrId &amp;attrId, int value)</code></td>
<td>Set a global int attribute</td>
</tr>
<tr>
<td><code>int GetInt (const Attr::IntAttrId &amp;attrId) const</code></td>
<td>Get a global int attribute</td>
</tr>
<tr>
<td><code>void SetFloat (const Attr::FloatAttrId &amp;attrId, float value)</code></td>
<td>Set a global float attribute</td>
</tr>
<tr>
<td><code>float GetFloat (const Attr::FloatAttrId &amp;attrId) const</code></td>
<td>Get a global float attribute</td>
</tr>
<tr>
<td><code>void SetBool (const Attr::BoolAttrId &amp;attrId, bool value)</code></td>
<td>Set a global bool attribute</td>
</tr>
<tr>
<td><code>bool GetBool (const Attr::BoolAttrId &amp;attrId) const</code></td>
<td>Get a global bool attribute</td>
</tr>
<tr>
<td><code>void SetFloat4 (const Attr::Float4AttrId &amp;attrId, const Math::float4 &amp;value)</code></td>
<td>Set a global float4 attribute</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Math::float4 GetFloat4</code> (const Attr::Float4AttrId &amp;attrId) const</td>
<td>set a global float4 attribute</td>
</tr>
<tr>
<td><code>void SetMatrix44</code> (const Attr::Matrix44AttrId &amp;attrId, const Math::matrix44 &amp;value)</td>
<td>get a global matrix44 attribute</td>
</tr>
<tr>
<td><code>Math::matrix44 GetMatrix44</code> (const Attr::Matrix44AttrId &amp;attrId) const</td>
<td>set a global matrix44 attribute</td>
</tr>
<tr>
<td><code>void SetGuid</code> (const Attr::GuidAttrId &amp;attrId, const Util::Guid &amp;guid)</td>
<td>set a global guid attribute</td>
</tr>
<tr>
<td><code>Util::Guid &amp; GetGuid</code> (const Attr::GuidAttrId &amp;attrId) const</td>
<td>get a global guid attribute</td>
</tr>
<tr>
<td><code>void SetBlob</code> (const Attr::BlobAttrId &amp;attrId, const Util::Blob &amp;blob)</td>
<td>set a global blob attribute</td>
</tr>
<tr>
<td><code>Util::Blob &amp; GetBlob</code> (const Attr::BlobAttrId &amp;attrId) const</td>
<td>get a global blob attribute</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> const</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>
</tbody>
</table>
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**)

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

**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</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
</table>

*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

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::CategoryManager, BaseGameFeature::EntityManager, BaseGameFeature::EnvEntityManager, and BaseGameFeature::EnvQueryManager.

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 \texttt{Messaging::Port}. 

\begin{verbatim}
void Messaging::Dispatcher::AttachPort( const\texttt{Ptr<Port> &port} ) [inherited]
\end{verbatim}

attach a message port

Attach a new message port.

\textbf{Parameters:}

\begin{itemize}
\item \textit{port} pointer to a message port object
\end{itemize}

\begin{verbatim}
void Messaging::Dispatcher::RemovePort( const\texttt{Ptr<Port> &port} ) [inherited]
\end{verbatim}

remove a message port

Remove a message port object.

\textbf{Parameters:}

\begin{itemize}
\item \textit{handler} pointer to message port object to be removed
\end{itemize}

\begin{verbatim}
bool Messaging::Dispatcher::HasPort( const\texttt{Ptr<Port> &port} ) const [inherited]
\end{verbatim}

return true if a port exists

Return true if a port is already attached.

\begin{verbatim}
void Messaging::Port::AttachHandler( const\texttt{Ptr<Handler> &h} ) [inherited]
\end{verbatim}

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

Increment the refcount of the object.
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 Tue Feb 19 12:16:43 2008
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

\begin{verbatim}
static bool Load (const Util::String &levelName)
load a complete level from the world database
\end{verbatim}
BaseGameFeature::LoaderServer
BaseGameFeature::LoaderServer Class Reference

#include <loaderserver.h>

Inheritance diagram for BaseGameFeature::LoaderServer:

```
Core::RefCounted

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><strong>LoaderServer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~LoaderServer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <strong>SetDebugTextEnabled</strong> (bool b)**</td>
<td>Enable/disable debug text messages during load</td>
</tr>
<tr>
<td>bool <strong>GetDebugTextEnabled</strong> () const</td>
<td>Get debug text enabled flag</td>
</tr>
<tr>
<td>virtual bool <strong>Open ()</strong></td>
<td>Open the loader subsystem</td>
</tr>
<tr>
<td>virtual void <strong>Close ()</strong></td>
<td>Close the loader subsystem</td>
</tr>
<tr>
<td>bool <strong>IsOpen ()</strong> const</td>
<td>Return true if open</td>
</tr>
<tr>
<td>virtual Ptr &amp; <strong>CreateUserProfile</strong> () const</td>
<td>Create a new user profile object</td>
</tr>
<tr>
<td>void <strong>SetUserProfile</strong> (const Ptr &amp; <strong>UserProfile</strong>) &amp;</td>
<td>Set the current user profile</td>
</tr>
<tr>
<td>const Ptr &amp; <strong>GetUserProfile</strong> () const</td>
<td>Get the current user profile</td>
</tr>
<tr>
<td>virtual bool <strong>LoadLevel</strong> (const Util::String &amp;levelName)</td>
<td>Load a new level, this method is usually called by Game::SetupManager</td>
</tr>
<tr>
<td>void <strong>AttachEntityLoader</strong> (const Ptr &amp; <strong>EntityLoaderBase</strong>) &amp;</td>
<td>Attach loader</td>
</tr>
<tr>
<td>void <strong>RemoveEntityLoader</strong> (const Ptr &amp; <strong>EntityLoaderBase</strong>) &amp;</td>
<td>Remove loader</td>
</tr>
<tr>
<td>void <strong>RemoveAllLoaders</strong> ()</td>
<td>Remove all loaders</td>
</tr>
<tr>
<td>void <strong>LoadEntities</strong> (const Util::Array &amp; <strong>Util::String</strong>) &amp;</td>
<td></td>
</tr>
</tbody>
</table>
&activeLayers)

load entities from db with entityloader

void SetProgressResource (const Util::String &r)
set progress indicator gui resource

const Util::String & GetProgressResource () const
get progress indicator gui resource

void SetMaxProgressValue (int v)
set the max progress value

int GetMaxProgressValue () const
get the max progress value

void AdvanceProgress (int amount)
advance the progress indicator

void SetProgressText (const Util::String &s)
set optional progress text

const Util::String & GetProgressText () const
get optional progress text

virtual void UpdateProgressDisplay ()
update the progress indicator display

void OpenProgressIndicator ()
open the progress indicator

void CloseProgressIndicator ()
close the progress indicator

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
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsA (const Rtti &amp;rtti) const</code></td>
<td>Check if 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>Check if 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>Check if 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>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>
Member Function Documentation

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

void
BaseGameFeature::LoaderServer::Close ( ) [virtual]

close the loader subsystem

Close the loader subsystem.

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.

void
BaseGameFeature::LoaderServer::SetUserProfile ( const Ptr< UserProfile > & p ) [inline]

set the current user profile

This sets the current user profile.

const Ptr< UserProfile > &
BaseGameFeature::LoaderServer::GetUserProfile ( ) const [inline]
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
const Util::FourCC &
```
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.
BaseGameFeature::MoveFollow
BaseGameFeature::MoveFollow Class Reference

#include <movefollow.h>

Inheritance diagram for BaseGameFeature::MoveFollow:
Detailed Description

A MoveFollow message. The expected behaviour is that the entity which receives this Message follows the target entity (defined by the target entity's unique id) until told otherwise.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MoveFollow</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td><strong>SetTargetEntityId</strong>(Game::Entity::EntityId id)</td>
<td>set the target entity's unique id</td>
</tr>
<tr>
<td><strong>GetTargetEntityId</strong>() const</td>
<td>get the target entity's unique id</td>
</tr>
<tr>
<td><strong>SetDistance</strong>(float d)</td>
<td>set the distance to keep to the target entity</td>
</tr>
<tr>
<td><strong>GetDistance</strong>() const</td>
<td>get the distance to keep to the target entity</td>
</tr>
<tr>
<td><strong>CheckId</strong>(const Messaging::Id &amp;id) const</td>
<td>return true if message is of the given id</td>
</tr>
<tr>
<td><strong>Encode</strong>(const Ptr<a href="">IO::BinaryWriter</a> &amp;writer)</td>
<td>encode message into a stream</td>
</tr>
<tr>
<td><strong>Decode</strong>(const Ptr<a href="">IO::BinaryReader</a> &amp;reader)</td>
<td>decode message from a stream</td>
</tr>
<tr>
<td><strong>SetHandled</strong>(bool b)</td>
<td>set the handled flag</td>
</tr>
<tr>
<td><strong>Handled</strong>() const</td>
<td>return true if the message has been handled</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)</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><strong>IsInstanceOf</strong>(const Util::FourCC)</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>bool &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.
BaseGameFeature::UserProfile
# BaseGameFeature::UserProfile Class Reference

#include <userprofile.h>

Inheritance diagram for BaseGameFeature::UserProfile:

```
Core::RefCounted

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><strong>UserProfile</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~UserProfile ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>void SetName (const Util::String &amp;n)</td>
<td>set the name of the user profile</td>
</tr>
<tr>
<td>const Util::String &amp; GetName () const</td>
<td>get the name of the user profile</td>
</tr>
<tr>
<td>virtual void SetToDefault ()</td>
<td>set the user profile to its default state, override in subclass</td>
</tr>
<tr>
<td>virtual bool Load (const Util::String &amp;path=&quot;&quot;)</td>
<td>load the profile data</td>
</tr>
<tr>
<td>virtual bool Save ()</td>
<td>save to disk</td>
</tr>
<tr>
<td>bool IsLoaded () const</td>
<td>currently loaded?</td>
</tr>
<tr>
<td>Util::String GetProfileDirectory () const</td>
<td>get the filesystem path to the user profile directory</td>
</tr>
<tr>
<td>Util::String GetSaveGameDirectory () const</td>
<td>get the filesystem path to the savegame directory</td>
</tr>
<tr>
<td>Util::String GetDatabasePath () const</td>
<td>get path to world database</td>
</tr>
<tr>
<td>Util::String GetSaveGamePath (const Util::String &amp;saveGameName) const</td>
<td>get path to a complete savegame</td>
</tr>
<tr>
<td>bool HasAttr (const Util::String &amp;name) const</td>
<td>return true if attribute exists in the profile</td>
</tr>
<tr>
<td>void SetString (const Util::String &amp;name, const Util::String &amp;val)</td>
<td>set a string attribute in the profile</td>
</tr>
<tr>
<td>void SetInt (const Util::String &amp;name, int val)</td>
<td>set an int attribute in the profile</td>
</tr>
<tr>
<td>void SetFloat (const Util::String &amp;name, float val)</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
```

```cpp
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
```

```cpp
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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static Util::Array &lt; Util::String &gt;</strong> EnumProfiles ()</td>
<td>static method to enumerate all existing user profiles</td>
</tr>
<tr>
<td><strong>static void DeleteProfile (const Util::String &amp; name)</strong></td>
<td>static method to delete an existing user profile by name</td>
</tr>
<tr>
<td><strong>static Util::String GetProfileRootDirectory ()</strong></td>
<td>static method which returns a path to the profile root directory</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>
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 overriden 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
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:

```
Core::RefCounted
  
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>virtual <code>~CoreServer ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <code>SetCompanyName (const Util::String &amp;s)</code></td>
<td>Set the company name</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetCompanyName ()</code> const</td>
<td>Get the company name</td>
</tr>
<tr>
<td>void <code>SetAppName (const Util::String &amp;s)</code></td>
<td>Set the application name</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetAppName ()</code> const</td>
<td>Get the application name</td>
</tr>
<tr>
<td>void <code>Open ()</code></td>
<td>Open the core server</td>
</tr>
<tr>
<td>void <code>Close ()</code></td>
<td>Close the core server</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 (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></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 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.
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>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>Register</code> (const <code>Rtti</code> *rtti, const <code>Util::String</code> &amp;className, const <code>Util::FourCC</code> &amp;classFourCC)</td>
<td>register a RTTI object with the factory</td>
</tr>
<tr>
<td>bool</td>
<td><code>ClassExists</code> (const <code>Util::String</code> &amp;className) const</td>
<td>check if a class exists by class name</td>
</tr>
<tr>
<td>bool</td>
<td><code>ClassExists</code> (const <code>Util::FourCC</code> classFourCC) const</td>
<td>check if a class exists by FourCC code</td>
</tr>
<tr>
<td>const <code>Rtti</code> *</td>
<td><code>GetClassRtti</code> (const <code>Util::String</code> &amp;className) const</td>
<td>get class rtti object by name</td>
</tr>
<tr>
<td>const <code>Rtti</code> *</td>
<td><code>GetClassRtti</code> (const <code>Util::FourCC</code> &amp;classFourCC) const</td>
<td>get class rtti object by fourcc code</td>
</tr>
<tr>
<td><code>RefCounted</code> *</td>
<td><code>Create</code> (const <code>Util::String</code> &amp;className) const</td>
<td>create an object by class name</td>
</tr>
<tr>
<td><code>RefCounted</code> *</td>
<td><code>Create</code> (const <code>Util::FourCC</code> classFourCC) const</td>
<td>create an object by FourCC code</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory *</td>
<td><strong>Instance ()</strong></td>
<td>get pointer to singleton instance (cannot use singleton.h!)</td>
</tr>
<tr>
<td>static void</td>
<td><strong>Destroy ()</strong></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 * rtti,
                              const Util::String & className,
                              const Util::FourCC & classFourCC
                          )
```

register a RTTI object with the factory

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 undefined).
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::Ptr
Core::Ptr Class Reference

#include <ptr.h>
Detailed Description

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
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.

(C) 2006 RadonLabs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RefCounted ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>int <strong>GetRefCount ()</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></td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong></td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong></td>
<td>return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool <strong>IsA</strong></td>
<td>return true if this object is instance of given class, or a derived class</td>
</tr>
<tr>
<td>bool <strong>IsA</strong></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>const <strong>Util::String &amp;</strong></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>GetClassName ()</strong></td>
<td>get the class name</td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></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

virtual ~RefCounted ()

destructor (called when refcount reaches zero)
Member Function Documentation

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

Return the current refcount of the object.
```

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

Increment the refcount of the object.
```

```
void Core::RefCounted::Release() [inline]
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]
get the class name

Get the class name of the object.
```

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

Get the class FourCC of the object.
```

```
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:

```plaintext
Util::List< TYPE >

Core::RefCountedList
```

< 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>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>DumpLeaks</strong> ()</td>
<td>Dump memory leaks, this method is called by <code>RefCounted::DumpRefCountedLeaks()</code></td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsEmpty</strong> () const</td>
<td>Return true if the list is empty</td>
</tr>
<tr>
<td>SizeT</td>
<td><strong>Size</strong> () const</td>
<td>Get number of elements in list (slow)</td>
</tr>
<tr>
<td>void</td>
<td><strong>Clear</strong> ()</td>
<td>Clear list</td>
</tr>
<tr>
<td>void</td>
<td><strong>AddList</strong> (const List&lt; TYPE &gt; &amp;l)</td>
<td>Add contents of other list to this list</td>
</tr>
<tr>
<td>Iterator</td>
<td><strong>AddAfter</strong> (Iterator iter, const TYPE &amp;e)</td>
<td>Add element after given element</td>
</tr>
<tr>
<td>Iterator</td>
<td><strong>AddBefore</strong> (Iterator iter, const TYPE &amp;e)</td>
<td>Add element before given element</td>
</tr>
<tr>
<td>Iterator</td>
<td><strong>AddFront</strong> (const TYPE &amp;e)</td>
<td>Add element to beginning of list</td>
</tr>
<tr>
<td>Iterator</td>
<td><strong>AddBack</strong> (const TYPE &amp;e)</td>
<td>Add element to end of list</td>
</tr>
<tr>
<td>TYPE</td>
<td><strong>RemoveFront</strong> ()</td>
<td>Remove first element of list</td>
</tr>
<tr>
<td>TYPE</td>
<td><strong>RemoveBack</strong> ()</td>
<td>Remove last element of list</td>
</tr>
<tr>
<td>TYPE</td>
<td><strong>Remove</strong> (Iterator iter)</td>
<td>Remove given element</td>
</tr>
<tr>
<td>TYPE &amp;</td>
<td><strong>Front</strong> () const</td>
<td>Get first element</td>
</tr>
<tr>
<td>TYPE &amp;</td>
<td><strong>Back</strong> () const</td>
<td>Get last element</td>
</tr>
<tr>
<td>Iterator</td>
<td><strong>Begin</strong> () const</td>
<td>Get iterator to first element</td>
</tr>
<tr>
<td>Iterator</td>
<td><strong>End</strong> () const</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
Core::Rtti Class Reference

#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, Util::FourCC fcc, Creator creatorFunc, const Core::Rtti *parentClass)</td>
<td>Constructor</td>
</tr>
<tr>
<td>bool operator==(const Rtti &amp;rhs) const</td>
<td>Equality operator</td>
</tr>
<tr>
<td>bool operator!=(const Rtti &amp;rhs) const</td>
<td>Inequality operator</td>
</tr>
<tr>
<td>const Util::String &amp; GetName () const</td>
<td>Get class name</td>
</tr>
<tr>
<td>Util::FourCC GetFourCC () const</td>
<td>Get four character code of class</td>
</tr>
<tr>
<td>const Core::Rtti * GetParent () const</td>
<td>Get pointer to parent class</td>
</tr>
<tr>
<td>RefCounted * Create () const</td>
<td>Create an object of this class</td>
</tr>
<tr>
<td>bool IsDerivedFrom (const Rtti &amp;other) const</td>
<td>Return true if this rtti is equal or derived from to other rtti</td>
</tr>
<tr>
<td>bool IsDerivedFrom (const Util::String &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 IsDerivedFrom (const Util::FourCC &amp;otherClassFourCC) const</td>
<td>Return true if this rtti is equal or derived from to other rtti, by fourcc</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:44 2008
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
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

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>
</table>

"enum"
## Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static Code_FromString</code> (const Util::String &amp;str)</td>
<td>convert adapter code from string</td>
</tr>
<tr>
<td><code>static Util::String_ToString</code> (Code code)</td>
<td>convert adapter code to string</td>
</tr>
</tbody>
</table>
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>SetDriverName (const Util::String &amp;s)</strong></td>
<td>set driver name</td>
</tr>
<tr>
<td><strong>GetDriverName () const</strong></td>
<td>get human readable driver name</td>
</tr>
<tr>
<td><strong>SetDescription (const Util::String &amp;s)</strong></td>
<td>set description string</td>
</tr>
<tr>
<td><strong>GetDescription () const</strong></td>
<td>get human readable description</td>
</tr>
<tr>
<td><strong>SetDeviceName (const Util::String &amp;s)</strong></td>
<td>set device name</td>
</tr>
<tr>
<td><strong>GetDeviceName () const</strong></td>
<td>get human readable device name</td>
</tr>
<tr>
<td><strong>SetDriverVersionLowPart (uint v)</strong></td>
<td>set driver version low part</td>
</tr>
<tr>
<td><strong>GetDriverVersionLowPart () const</strong></td>
<td>get low part of driver version</td>
</tr>
<tr>
<td><strong>SetDriverVersionHighPart (uint v)</strong></td>
<td>set driver version high part</td>
</tr>
<tr>
<td><strong>GetDriverVersionHighPart () const</strong></td>
<td>get high part of driver version</td>
</tr>
<tr>
<td><strong>SetVendorId (uint id)</strong></td>
<td>set vendor id</td>
</tr>
<tr>
<td><strong>GetVendorId () const</strong></td>
<td>get vendor identifier</td>
</tr>
<tr>
<td><strong>SetDeviceId (uint id)</strong></td>
<td>set device id</td>
</tr>
<tr>
<td><strong>GetDeviceId () const</strong></td>
<td>get device identifier</td>
</tr>
<tr>
<td><strong>SetSubSystemId (uint id)</strong></td>
<td>set subsystem id</td>
</tr>
<tr>
<td><strong>GetSubSystemId () const</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void SetRevision(uint r)</code></td>
<td>set hardware revision</td>
</tr>
<tr>
<td><code>uint GetRevision()</code></td>
<td>get hardware revision identifier</td>
</tr>
<tr>
<td><code>void SetGuid(const Util::Guid &amp;g)</code></td>
<td>set driver/chipset pair guid</td>
</tr>
<tr>
<td><code>const Util::Guid &amp; GetGuid()</code></td>
<td>get guid for driver/chipset pair</td>
</tr>
</tbody>
</table>
CoreGraphics::AntiAliasQuality
CoreGraphics::AntiAliasQuality Class Reference

#include <antialiasquality.h>
Detailed Description

Anti-alias quality levels.

(C) 2006 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Public Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum Code</td>
</tr>
<tr>
<td>enum</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>static <code>Code</code></th>
<th><code>FromString</code> (const <code>Util::String</code> &amp;str)</th>
<th>convert from string</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Util::String</code></td>
<td><code>ToString</code> (<code>Code</code> code)</td>
<td>convert to string</td>
</tr>
</tbody>
</table>
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>
</tr>
</thead>
</table>

(batch type enum)
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static BatchType::Code FromString (const Util::String &amp;str)</code></td>
<td>convert from string</td>
</tr>
<tr>
<td><code>static Util::String ToString (BatchType::Code c)</code></td>
<td>convert to string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:44 2008
CoreGraphics::CPUIndexBuffer
CoreGraphics::CPUIndexBuffer Class Reference

#include <cpuindexbuffer.h>

Inheritance diagram for CoreGraphics::CPUIndexBuffer:
Detailed Description

CPU 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>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPUIndexBuffer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~CPUIndexBuffer ()</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>void * <strong>Map (MapType mapType)</strong></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>SetIndexBufferDepth (CoreGraphics::IndexType::Code depth)</strong></td>
<td>Set index buffer depth, to allow changing depth</td>
</tr>
<tr>
<td>CoreGraphics::IndexType::Code <strong>GetIndexBufferDepth () const</strong></td>
<td>Get index buffer depth</td>
</tr>
<tr>
<td>CoreGraphics::IndexType::Code <strong>GetIndexType () const</strong></td>
<td>Get the index type (Index16 or Index32)</td>
</tr>
<tr>
<td>SizeT <strong>GetNumIndices () const</strong></td>
<td>Get number of indices</td>
</tr>
<tr>
<td>Usage <strong>GetUsage () const</strong></td>
<td>Get resource usage type</td>
</tr>
<tr>
<td>Access <strong>GetAccess () const</strong></td>
<td>Get cpu access type</td>
</tr>
<tr>
<td>void <strong>SetAsyncEnabled (bool b)</strong></td>
<td>Request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td>bool <strong>IsAsyncEnabled () const</strong></td>
<td>Return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td>void <strong>SetResourceId (const ResourceId &amp;id)</strong></td>
<td>Set the resource identifier</td>
</tr>
<tr>
<td>const ResourceId &amp; <strong>GetResourceId () const</strong></td>
<td>Get the resource identifier</td>
</tr>
</tbody>
</table>
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

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
<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>
</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>SetIndexType (CoreGraphics::IndexType::Code type)</code></td>
<td>set the index type (Index16 or Index32)</td>
</tr>
<tr>
<td><code>SetNumIndices (SizeT num)</code></td>
<td>set number of indices</td>
</tr>
<tr>
<td><code>SetUsage (Usage usage)</code></td>
<td>set resource usage type</td>
</tr>
<tr>
<td><code>SetAccess (Access access)</code></td>
<td>set resource cpu access type</td>
</tr>
<tr>
<td><code>SetState (State s)</code></td>
<td>set current state</td>
</tr>
<tr>
<td><code>IncrUseCount ()</code></td>
<td>increment use count</td>
</tr>
<tr>
<td><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.

**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.

### Get the class name

```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

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

Get the class FourCC code

Get the class FourCC of the object.

### Dump refcounting leaks

```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::CPUMemoryIndexBufferLoader
CoreGraphics::CPUMemoryIndexBufferLoader

#include <cpumemoryindexbufferloader.h>

Inheritance diagram for
CoreGraphics::CPUMemoryIndexBufferLoader:
Detailed Description

Initialize a D3D9IndexBuffer from data in memory. 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>Function</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 <strong>Setup</strong> (CoreGraphics::IndexType::Code indexType, SizeT numIndices, void *ptr, SizeT numBytes)</td>
<td>setup index buffer data, must remain valid until <strong>OnLoadRequested</strong>() is called!</td>
</tr>
<tr>
<td>void <strong>Setup</strong> (CoreGraphics::IndexType::Code indexType, SizeT numIndices, SizeT numBytes, CoreGraphics::IndexBuffer::Usage usage, CoreGraphics::IndexBuffer::Access access)</td>
<td>setup a empty index buffer</td>
</tr>
<tr>
<td>virtual void <strong>OnAttachToResource</strong> (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 <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>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetState () const</code></td>
<td>return current 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><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 (NEBUŁA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
void SetState (Resource::State S)  
set current state
```
Member Function Documentation

bool
CoreGraphics::CPUMemoryIndexBufferLoader::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). The data pointer provided to **Setup()** will be invalidated inside **OnLoadRequested()**. Usage will be set to UsageImmutable and Access to AccessNone.

Reimplemented from **Resources::ResourceLoader**.

```cpp
void Base::MemoryIndexBufferLoaderBase::Setup( CoreGraphics::IndexType::Code indexType,
   SizeT num,
   void * ptr,
   SizeT numBytes )
```

setup index buffer data, must remain valid until **OnLoadRequested()** is called!

Setup all information needed to initialize the IndexBuffer resource. The data must remain valid until **OnLoadRequested()** is called (which will invalidate the data).

```cpp
void Base::MemoryIndexBufferLoaderBase::Setup( CoreGraphics::IndexType::Code indexType,
   SizeT num,
   SizeT numBytes,
   CoreGraphics::IndexBuffer::Usage usage,
   CoreGraphics::IndexBuffer::Access access )
```

setup a empty index buffer

Setup all information needed to initialize a empty IndexBuffer
setup index buffer data, must remain valid until **OnLoadRequested()** is called!

Setup all information needed to initialize a IndexBuffer resource.

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**, **Direct3D9::D3D9StreamTextureLoader**, **CoreGraphics::StreamAnimationLoader**, **CoreGraphics::StreamMeshLoader**, and **Models::StreamModelLoader**.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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::CPUMemoryVertexBufferLoader
CoreGraphics::CPUMemoryVertexBufferLoader Class Reference

#include <cpumemoryvertexbufferloader.h>

Inheritance diagram for 
CoreGraphics::CPUMemoryVertexBufferLoader:

```
<table>
<thead>
<tr>
<th>Core::RefCounted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources::ResourceLoader</td>
</tr>
<tr>
<td>Base::MemoryVertexBufferLoaderBase</td>
</tr>
<tr>
<td>CoreGraphics::CPUMemoryVertexBufferLoader</td>
</tr>
</tbody>
</table>
```
Detailed Description

Initialize a vertex buffer blob from data in memory. This resource loader creates VertexBuffers which are read and writeable 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 <code>OnLoadRequested()</code></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td>void <code>Setup</code> (const <code>Util::Array&lt;CoreGraphics::VertexComponent&gt;</code> &amp;vertexComponents, SizeT numVertices, void *ptr, SizeT numBytes)</td>
<td>setup vertex buffer data, must remain valid until <code>OnLoadRequested()</code> is called!</td>
</tr>
<tr>
<td>void <code>Setup</code> (const <code>Util::Array&lt;CoreGraphics::VertexComponent&gt;</code> &amp;vertexComponents, SizeT numVertices, SizeT numBytes, <code>CoreGraphics::VertexBuffer::Usage</code> usage, <code>CoreGraphics::VertexBuffer::Access</code> access)</td>
<td>setup a vertex buffer, vertex buffer data, must remain valid until <code>OnLoadRequested()</code> is called!</td>
</tr>
<tr>
<td>virtual void <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>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 <code>Ptr&lt;Resource&gt;</code> &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>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</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>Resource::State GetState() const</strong></td>
<td>return current state</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!)
Protected Member Functions

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

set current state
```
Member Function Documentation

```cpp
bool CoreGraphics::CPUMemoryVertexBufferLoader::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). The data pointer provided to Setup() will be invalidated inside OnLoadRequested(). Resource usage will be set to UsageImmutable and resource access to AccessNone.

Reimplemented from Resources::ResourceLoader.

```cpp
void Base::MemoryVertexBufferLoaderBase::Setup( const Util::Array< CoreGraphics::VertexComponent > & components,
                                                SizeT num,
                                                void * ptr,
                                                SizeT numBytes )

```cpp
void Base::MemoryVertexBufferLoaderBase::Setup( const Util::Array< CoreGraphics::VertexComponent > & vertexComponents,
                                                SizeT num,
                                                SizeT numBytes,
                                                CoreGraphics::VertexBuffer::Usage usage,
                                                CoreGraphics::VertexBuffer::Access access )

```cpp
setup vertex buffer data, must remain valid until OnLoadRequested() is called!

Setup all information needed to initialize the VertexBuffer resource. The data must remain valid until OnLoadRequested() is called (which will invalidate the data).
```
setup a empty vertex buffer

Setup all information needed to initialize a empty VertexBuffer resource.

```cpp
void Base::MemoryVertexBufferLoaderBase::Setup(
    const Util::Array<CoreGraphics::VertexComponent>& components,
    SizeT num, void* ptr, SizeT numBytes,
    CoreGraphics::VertexBuffer::Usage usage,
    CoreGraphics::VertexBuffer::Access access)
```

setup a vertex buffer, vertex buffer data, must remain valid until `OnLoadRequested()` is called!

Setup all information needed to initialize the VertexBuffer resource. The data must remain valid until `OnLoadRequested()` is called (which will invaliate the data).

```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`, `Direct3D9::D3D9StreamTextureLoader`, `CoreGraphics::StreamAnimationLoader`, `CoreGraphics::StreamMeshLoader`, and `Models::StreamModelLoader`. 
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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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::CPUVertexBuffer
CoreGraphics::CPUVertexBuffer Class Reference

#include <cpuvertexbuffer.h>

Inheritance diagram for CoreGraphics::CPUVertexBuffer:
Detailed Description

Implementation of a CPU VertexBuffer hold in main memory.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>Usage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>resource usage flags</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>State</strong></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>CPUVertexBuffer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~CPUVertexBuffer ()</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>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>const <strong>Ptr&lt; CoreGraphics::VertexLayout &gt; &amp;</strong></td>
<td><strong>GetVertexLayout</strong> ( ) const</td>
</tr>
<tr>
<td></td>
<td>Get the vertex layout</td>
</tr>
<tr>
<td><strong>GetSizeT</strong> ( ) const</td>
<td>Get number of vertices in the buffer</td>
</tr>
<tr>
<td><strong>GetUsage</strong> ( ) const</td>
<td>Get resource usage type</td>
</tr>
<tr>
<td><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>void <strong>SetResourceId</strong> (const Resourceld &amp;id)</td>
<td>Set the resource identifier</td>
</tr>
<tr>
<td>const <strong>Resourceld &amp;</strong></td>
<td><strong>GetResourceId</strong> ( ) const</td>
</tr>
<tr>
<td></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>const <strong>Ptr&lt; ResourceLoader &gt; &amp;</strong></td>
<td><strong>GetLoader</strong> ( ) const</td>
</tr>
</tbody>
</table>
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

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 ()
decomment 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 <strong>Rtti</strong> &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>
<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>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>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetVertexLayout</strong> (const Ptr&lt; CoreGraphics::VertexLayout &gt; &amp;vertexLayout)</td>
<td>set vertex layout (set by resource loader)</td>
</tr>
<tr>
<td><strong>SetNumVertices</strong> (SizeT numVertices)</td>
<td>set number of vertices (set by resource loader)</td>
</tr>
<tr>
<td><strong>SetUsage</strong> (Usage usage)</td>
<td>set resource usage type</td>
</tr>
<tr>
<td><strong>SetAccess</strong> (Access access)</td>
<td>set resource cpu access type</td>
</tr>
<tr>
<td><strong>SetState</strong> (State s)</td>
<td>set current state</td>
</tr>
<tr>
<td><strong>IncrUseCount</strong> ()</td>
<td>increment use count</td>
</tr>
<tr>
<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.
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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DisplayDevice ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~DisplayDevice ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>AdapterExists (CoreGraphics::Adapter::Code adapter)</strong></td>
<td>check if the adapter actually exists</td>
</tr>
<tr>
<td><strong>GetAvailableDisplayModes (CoreGraphics::Adapter::Code adapter, CoreGraphics::PixelFormat::Code pixelFormat)</strong></td>
<td>get available display modes on given adapter</td>
</tr>
<tr>
<td><strong>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>GetCurrentAdapterDisplayMode (CoreGraphics::Adapter::Code adapter)</strong></td>
<td>get current adapter display mode (i.e. the desktop display mode)</td>
</tr>
<tr>
<td><strong>GetAdapterInfo (CoreGraphics::Adapter::Code adapter)</strong></td>
<td>get general info about display adapter</td>
</tr>
<tr>
<td>virtual bool <strong>Open ()</strong></td>
<td>open the display</td>
</tr>
<tr>
<td>virtual void <strong>Close ()</strong></td>
<td>close the display</td>
</tr>
<tr>
<td>virtual void <strong>ProcessWindowMessages ()</strong></td>
<td>process window system messages, call this method once per frame</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>HWND GetHwnd() const</td>
<td>get the application window HWND</td>
</tr>
<tr>
<td>void SetAdapter(CoreGraphics::Adapter::Code a)</td>
<td>set display adapter (make sure adapter exists!)</td>
</tr>
<tr>
<td>CoreGraphics::Adapter::Code GetAdapter() const</td>
<td>get display adapter</td>
</tr>
<tr>
<td>void SetDisplayMode(const CoreGraphics::DisplayMode &amp;m)</td>
<td>set display mode (make sure the display mode is supported!)</td>
</tr>
<tr>
<td>const CoreGraphics::DisplayMode &amp; GetDisplayMode() const</td>
<td>get display mode</td>
</tr>
<tr>
<td>void SetAntiAliasQuality(CoreGraphics::AntiAliasQuality::Code aa)</td>
<td>set antialias quality</td>
</tr>
<tr>
<td>CoreGraphics::AntiAliasQuality::Code GetAntiAliasQuality() const</td>
<td>get antialias quality</td>
</tr>
<tr>
<td>void SetFullscreen(bool b)</td>
<td>set windowed/fullscreen mode</td>
</tr>
<tr>
<td>bool IsFullscreen() const</td>
<td>get windowed/fullscreen mode</td>
</tr>
<tr>
<td>void SetDisplayModeSwitchEnabled(bool b)</td>
<td>enable display mode switch when running fullscreen (default is true);</td>
</tr>
<tr>
<td>bool IsDisplayModeSwitchEnabled() const</td>
<td>is display mode switch enabled for fullscreen?</td>
</tr>
<tr>
<td>void SetTripleBufferingEnabled(bool b)</td>
<td>enable triple buffer for fullscreen (default is double buffering)</td>
</tr>
<tr>
<td>bool IsTripleBufferingEnabled() const</td>
<td>is triple buffer enabled for fullscreen?</td>
</tr>
<tr>
<td>void SetAlwaysOnTop(bool b)</td>
<td>set always-on-top behaviour</td>
</tr>
<tr>
<td>bool IsAlwaysOnTop() const</td>
<td>get always-on-top behaviour</td>
</tr>
</tbody>
</table>
void SetVerticalSyncEnabled (bool b)
    turn vertical sync on/off

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 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 string

bool IsInstanceOf (const Util::FourCC &classFourCC) const
    return true if this object is instance of given class by
<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>
<tr>
<td>bool</td>
<td><code>IsA</code> (const <code>Util::String</code> &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td><code>return true if this object is instance of given class, or a derived class, by string</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</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 <code>Util::String</code> &amp;</td>
<td><code>GetClassName</code> () const</td>
</tr>
<tr>
<td></td>
<td><code>get the class name</code></td>
</tr>
<tr>
<td><code>Util::FourCC</code></td>
<td><code>GetClassFourCC</code> () const</td>
</tr>
<tr>
<td></td>
<td><code>get the class FourCC code</code></td>
</tr>
<tr>
<td>static void</td>
<td>DumpRefCountingLeaks ()</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------</td>
</tr>
</tbody>
</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 bool OpenWindow</td>
<td><code>open the application window</code></td>
</tr>
<tr>
<td>virtual void CloseWindow</td>
<td><code>close the application window</code></td>
</tr>
<tr>
<td>virtual void OnMinimized</td>
<td><code>called on WM_SIZE when window is minimized</code></td>
</tr>
<tr>
<td>virtual void OnRestored</td>
<td><code>called on WM_SIZE when window is restored</code></td>
</tr>
<tr>
<td>virtual bool OnSetCursor</td>
<td><code>called on WM_SETCURSOR</code></td>
</tr>
<tr>
<td>virtual void OnPaint</td>
<td><code>called on WM_PAINT</code></td>
</tr>
<tr>
<td>virtual void OnSetFocus</td>
<td><code>called on WM_SETFOCUS</code></td>
</tr>
<tr>
<td>virtual void OnKillFocus</td>
<td><code>called on WM_KILLFOCUS</code></td>
</tr>
<tr>
<td>virtual void OnCloseRequested</td>
<td><code>called on WM_CLOSE to request if window should be closed</code></td>
</tr>
<tr>
<td>virtual void OnToggleFullscreenWindowed</td>
<td><code>called when Alt-Enter is pressed</code></td>
</tr>
<tr>
<td>virtual void OnKeyDown</td>
<td><code>called on WM_KEYDOWN</code></td>
</tr>
<tr>
<td>virtual void OnKeyUp</td>
<td><code>called on WM_KEYUP</code></td>
</tr>
<tr>
<td>virtual void OnChar</td>
<td><code>called on WM_CHAR</code></td>
</tr>
<tr>
<td>virtual void OnMouseButton</td>
<td><code>called on mouse button event</code></td>
</tr>
<tr>
<td>virtual void OnMouseMove</td>
<td><code>called on WM_MOUSEMOVE</code></td>
</tr>
<tr>
<td>virtual void OnMouseWheel</td>
<td><code>called on WM_MOUSEWHEEL</code></td>
</tr>
</tbody>
</table>

**Input::Key::Code**

Translating KeyCode

(WPARAM wParam) const
*translate a Windows virtual key code into a Nebula3 key code*

<table>
<thead>
<tr>
<th>Math::float2</th>
<th><strong>ComputeAbsMousePos</strong> (LPARAM lParam) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>compute absolute mouse position from lParam</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Math::float2</th>
<th><strong>ComputeNormMousePos</strong> (const Math::float2 &amp;absMousePos) const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>compute normalized mouse position from absolute mouse pos</td>
</tr>
</tbody>
</table>

<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>
### 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>the WinProc</td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

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.

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.

```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 Ptr<CoreGraphics::DisplayEventHandler> & h) [i
```

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]

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.

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:44 2008
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
## Public Types

<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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DisplayEvent ()</strong></td>
<td>default constructor</td>
</tr>
<tr>
<td><strong>DisplayEvent (Code c)</strong></td>
<td>constructor with event code</td>
</tr>
<tr>
<td><strong>DisplayEvent (Code c, const Math::float2 &amp;absPos, const Math::float2 &amp;normPos)</strong></td>
<td>constructor with event code and mouse pos</td>
</tr>
<tr>
<td><strong>DisplayEvent (Code c, Input::Key::Code k)</strong></td>
<td>constructor with key code</td>
</tr>
<tr>
<td><strong>DisplayEvent (Code c, Input::Char chr)</strong></td>
<td>constructor with character</td>
</tr>
<tr>
<td><strong>DisplayEvent (Code c, Input::MouseButton::Code b, const Math::float2 &amp;absPos, const Math::float2 &amp;normPos)</strong></td>
<td>constructor with mouse button and mouse pos</td>
</tr>
</tbody>
</table>

#### Code

- **GetEventCode () const**
  - get event code
- **GetAbsMousePos () const**
  - get absolute mouse pos (in pixels)
- **GetNormMousePos () const**
  - get normalized mouse pos (from 0.0 to 1.0)
- **Input::Key::Code GetKey () const**
  - get key code
- **Input::Char GetChar () const**
  - get character code
- **Input::MouseButton::Code GetMouseButton () const**
  - get mouse button code
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 ~DisplayEventHandler ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void OnAttach ()</td>
<td>called when the event handler is attached to the DisplayDevice</td>
</tr>
<tr>
<td>virtual void OnRemove ()</td>
<td>called when the event handler is removed from the DisplayDevice</td>
</tr>
<tr>
<td>virtual bool PutEvent (const DisplayEvent &amp;event)</td>
<td>called by DisplayDevice when an event happens</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>
</tbody>
</table>

---

**Public Member Functions**

- **DisplayEventHandler ()**
  
  *constructor*

- virtual ~DisplayEventHandler ()
  
  *destructor*

- virtual void OnAttach ()
  
  *called when the event handler is attached to the DisplayDevice*

- virtual void OnRemove ()
  
  *called when the event handler is removed from the DisplayDevice*

- virtual bool PutEvent (const DisplayEvent &event)
  
  *called by DisplayDevice when an event happens*

- 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>
## Protected Member Functions

<table>
<thead>
<tr>
<th>virtual bool</th>
<th><strong>EventHandler</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.
```

void
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::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>const equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const DisplayMode &amp;rhs)</code></td>
<td>const 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>
float GetAspectRatio () const

get aspect ratio, this is simply width / height
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.
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><strong>Code</strong></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 (const Util::String &amp;str)</code></td>
<td>convert from string</td>
</tr>
<tr>
<td><code>ToString (Code c)</code></td>
<td>convert to string</td>
</tr>
<tr>
<td><code>FromMediaType (const IO::MediaType &amp;mediaType)</code></td>
<td>convert from media type (MIME)</td>
</tr>
<tr>
<td><code>ToMediaType (Code c)</code></td>
<td>convert to media type (MIME)</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:44 2008
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>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>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>IDirect3DIndexBuffer9 * <strong>GetD3D9IndexBuffer</strong> () const</td>
<td>get d3d9 index buffer pointer</td>
</tr>
<tr>
<td>CoreGraphics::IndexType::Code <strong>GetIndexType</strong> () const</td>
<td>get the index type (Index16 or Index32)</td>
</tr>
<tr>
<td>SizeT <strong>GetNumIndices</strong> () const</td>
<td>get number of indices</td>
</tr>
<tr>
<td><strong>GetUsage</strong> () const</td>
<td>get resource usage type</td>
</tr>
<tr>
<td><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>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>const Ptr&lt;ResourceLoader&gt; &amp; <strong>GetLoader</strong> () const</td>
<td>get optional resource loader</td>
</tr>
<tr>
<td>void <strong>SetSaver</strong> (const Ptr&lt;ResourceSaver&gt; &amp;saver)</td>
<td></td>
</tr>
</tbody>
</table>


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

SizeT GetUseCount () const
get current use count

virtual State Load ()
load the resource

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
<table>
<thead>
<tr>
<th><strong>IsA</strong> (const <strong>Util::String</strong> &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>
</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 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><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>void</th>
<th><strong>SetIndexType</strong> <em>(CoreGraphics::IndexType::Code</em> <em>type)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>set the index type <em>(Index16 or Index32)</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetNumIndices</strong> <em>(SizeT</em> <em>num)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>set number of indices</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetUsage</strong> <em>(Usage</em> <em>usage)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>set resource usage type</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetAccess</strong> <em>(Access</em> <em>access)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>set resource cpu access type</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetState</strong> <em>(State</em> <em>s)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>set current state</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>IncrUseCount</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>increment use count</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>DecrUseCount</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>decrement use count</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.

\texttt{const Util::String} &
\texttt{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.
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>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>index types enum</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static SizeT <strong>SizeOf</strong> (IndexType::Code type)</td>
<td>get byte size of index</td>
</tr>
<tr>
<td>static Util::String <strong>ToString</strong> (IndexType::Code type)</td>
<td>convert index type to string</td>
</tr>
<tr>
<td>static IndexType::Code <strong>FromString</strong> (const Util::String &amp;str)</td>
<td>convert string to index type</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Tue Feb 19 12:16:44 2008
CoreGraphics::MemoryIndexBufferLoader
#include <memoryindexbufferloader.h>

Inheritance diagram for 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><code>virtual bool OnLoadRequested()</code></td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td><code>void Setup (CoreGraphics::IndexType::Code indexType, SizeT numIndices, void *ptr, SizeT numBytes)</code></td>
<td>setup index buffer data, must remain valid until <code>OnLoadRequested()</code> is called!</td>
</tr>
<tr>
<td><code>void Setup (CoreGraphics::IndexType::Code indexType, SizeT numIndices, SizeT numBytes, CoreGraphics::IndexBuffer::Usage usage, CoreGraphics::IndexBuffer::Access access)</code></td>
<td>setup a empty index buffer</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>
</tbody>
</table>

Resource::State
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetState</strong>() const</td>
<td>return current 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</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>DumpRefCountingLeaks ()</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

bool Direct3D9::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). The data pointer provided to Setup() will be invalidated inside OnLoadRequested(). Usage will be set to UsageImmutable and Access to AccessNone.

Reimplemented from Resources::ResourceLoader.

void Base::MemoryIndexBufferLoaderBase::Setup(CoreGraphics::IndexType::Code indexType,
                                           SizeT num,
                                           void *ptr,
                                           SizeT numBytes)
setup index buffer data, must remain valid until OnLoadRequested() is called!

Setup all information needed to initialize the IndexBuffer resource. The data must remain valid until OnLoadRequested() is called (which will invalidate the data).

void Base::MemoryIndexBufferLoaderBase::Setup(CoreGraphics::IndexType::Code indexType,
                                           SizeT num,
                                           SizeT numBytes,
                                           CoreGraphics::IndexBuffer::Usage usage,
                                           CoreGraphics::IndexBuffer::Access access)
setup a empty index buffer

Setup all information needed to initialize a empty IndexBuffer
void Base::MemoryIndexBufferLoaderBase::Setup(CoreGraphics::IndexType::Code type, SizeT num, void *ptr, SizeT numBytes, CoreGraphics::IndexBuffer::Usage usage, CoreGraphics::IndexBuffer::Access access)

setup index buffer data, must remain valid until **OnLoadRequested()** is called!

Setup all information needed to initialize a IndexBuffer resource.

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**, **Direct3D9::D3D9StreamTextureLoader**, **CoreGraphics::StreamAnimationLoader**, **CoreGraphics::StreamMeshLoader**, and **Models::StreamModelLoader**.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

```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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

```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.
CoreGraphics::MemoryVertexBufferLoader
CoreGraphics::MemoryVertexBufferLoader
Class Reference

#include <memoryvertexbufferloader.h>

Inheritance diagram for CoreGraphics::MemoryVertexBufferLoader:

[Diagram showing inheritance hierarchy]

Core::RefCounted

Resources::ResourceLoader

Base::MemoryVertexBufferLoaderBase

Direct3DS::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 Calls</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</code> (const <code>Util::Array&lt;CoreGraphics::VertexComponent&gt;</code> &amp;vertexComponents, SizeT numVertices, void *ptr, SizeT numBytes)</td>
<td>Setup vertex buffer data, must remain valid until <code>OnLoadRequested()</code> is called!</td>
</tr>
<tr>
<td>void <code>Setup</code> (const <code>Util::Array&lt;CoreGraphics::VertexComponent&gt;</code> &amp;vertexComponents, SizeT numVertices, SizeT numBytes, <code>CoreGraphics::VertexBuffer::Usage</code> usage, <code>CoreGraphics::VertexBuffer::Access</code> access)</td>
<td>Setup a empty vertex buffer.</td>
</tr>
<tr>
<td>void <code>Setup</code> (const <code>Util::Array&lt;CoreGraphics::VertexComponent&gt;</code> &amp;vertexComponents, SizeT numVertices, void *ptr, SizeT numBytes, <code>CoreGraphics::VertexBuffer::Usage</code> usage, <code>CoreGraphics::VertexBuffer::Access</code> access)</td>
<td>Setup a vertex buffer, vertex buffer data, must remain valid until <code>OnLoadRequested()</code> is called!</td>
</tr>
<tr>
<td>virtual void <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>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 <code>Ptr&lt; Resource &gt;</code> &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>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</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>Resource::State GetState() const</td>
<td>return current 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>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

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

set current state
Member Function Documentation

bool Direct3D9::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). The data pointer provided to Setup() will be invalidated inside OnLoadRequested(). Resource usage will be set to UsageImmutable and resource access to AccessNone.

Reimplemented from Resources::ResourceLoader.

void Base::MemoryVertexBufferLoaderBase::Setup(const Util::Array<CoreGraphics::VertexComponent> & components, SizeT num, void * ptr, SizeT numBytes)

setup vertex buffer data, must remain valid until OnLoadRequested() is called!

Setup all information needed to initialize the VertexBuffer resource. The data must remain valid until OnLoadRequested() is called (which will invalidate the data).

void Base::MemoryVertexBufferLoaderBase::Setup(const Util::Array<CoreGraphics::VertexComponent> & vertexComponents, SizeT num, SizeT numBytes, CoreGraphics::VertexBuffer::Usage usage, CoreGraphics::VertexBuffer::Access access)
setup a empty vertex buffer

Setup all information needed to initialize a empty VertexBuffer resource.

```cpp
void Base::MemoryVertexBufferLoaderBase::Setup(
  const Util::Array<CoreGraphics::VertexComponent>& components,
  SizeT num, void* ptr, SizeT numBytes,
  CoreGraphics::VertexBuffer::Usage usage,
  CoreGraphics::VertexBuffer::Access access)
```

setup a vertex buffer, vertex buffer data, must remain valid until
**OnLoadRequested()** is called!

Setup all information needed to initialize the VertexBuffer resource.
The data must remain valid until **OnLoadRequested()** is called (which
will invalidate the data).

```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**, **Direct3D9::D3D9StreamTextureLoader**, **CoreGraphics::StreamAnimationLoader**, **CoreGraphics::StreamMeshLoader**, and **Models::StreamModelLoader**.
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 **Direct3D9::D3D9StreamTextureLoader**, **CoreGraphics::StreamAnimationLoader**, **CoreGraphics::StreamMeshLoader**, and **Models::StreamModelLoader**.

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 **Direct3D9::D3D9StreamTextureLoader**, **CoreGraphics::StreamAnimationLoader**, **CoreGraphics::StreamMeshLoader**, and **Models::StreamModelLoader**.

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:
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>
<tbody>
<tr>
<td></td>
<td>resource states (DO NOT CHANGE ORDER!)</td>
</tr>
</tbody>
</table>
Public Member Functions

virtual void **Unload** ()
*unload mesh resource*

bool **HasVertexBuffer** () const
*return true if the mesh has a vertex buffer*

const **Ptr** & < CoreGraphics::VertexBuffer > & **GetVertexBuffer** () const
*get the vertex buffer object*

bool **HasIndexBuffer** () const
*return true if the mesh has an index buffer*

const **Ptr** & < CoreGraphics::IndexBuffer > & **GetIndexBuffer** () const
*get the index buffer object*

**GetSizeT** **GetNumPrimitiveGroups** () const
*get the number of primitive groups in the mesh*

const CoreGraphics::PrimitiveGroup & **GetPrimitiveGroupAtIndex** (IndexT i)
*const*

virtual void **ApplyPrimitives** (IndexT primGroupIndex)
*apply any nesseccary mesh data in renderdevice*

void **SetPrimitiveGroups** (const Util::Array< CoreGraphics::PrimitiveGroup > &groups)
*set primitive groups*

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

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

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

const ResourceId & **GetResourceId** () const
*get the resource identifier*
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>State GetState () const</code></td>
<td>get current state</td>
</tr>
<tr>
<td><code>bool IsLoaded () const</code></td>
<td>return true if current state is Loaded</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</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) 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

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetVertexBuffer</strong> (const <em>Ptr</em>&lt; <em>CoreGraphics::VertexBuffer</em> &gt; &amp;vb)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set the vertex buffer object</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetIndexBuffer</strong> (const <em>Ptr</em>&lt; <em>CoreGraphics::IndexBuffer</em> &gt; &amp;ib)</td>
</tr>
<tr>
<td></td>
<td>set the index buffer object</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetState</strong> (<em>State</em> s)</td>
</tr>
<tr>
<td></td>
<td>set current state</td>
</tr>
<tr>
<td>void</td>
<td><strong>IncrUseCount</strong> ()</td>
</tr>
<tr>
<td></td>
<td>increment use count</td>
</tr>
<tr>
<td>void</td>
<td><strong>DecrUseCount</strong> ()</td>
</tr>
<tr>
<td></td>
<td>decrement use count</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Resource::State**

```cpp
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::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>enums</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>

- **FromString (const Util::String &str)**
- **ToString (Code code)**
Member Function Documentation

**PixelFormat::Code**
CoreGraphics::PixelFormat::FromStr\( \text{const} \ \text{Util::String} \ \text{str} \ ) \text{[static]}\n
convert from string

Convert a pixel format string into a pixel format code.

**Util::String**
CoreGraphics::PixelFormat::ToString\( \text{PixelFormat::Code} \ \text{code} \ ) \text{[static]}\n
convert to string

Convert pixel format code into a string.

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:44 2008
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

**PrimitiveGroup** ()
- constructor

void **SetBaseVertex** (IndexT i)
- set base vertex index

IndexT **GetBaseVertex** () const
- get index of base vertex

void **SetNumVertices** (SizeT n)
- set number of vertices

SizeT **GetNumVertices** () const
- get number of vertices

void **SetBaseIndex** (IndexT i)
- set base index index

IndexT **GetBaseIndex** () const
- get base index index

void **SetNumIndices** (SizeT n)
- set number of indices

SizeT **GetNumIndices** () const
- get number of indices

void **SetPrimitiveTopology** (**PrimitiveTopology::Code** topology)
- set the primitive topology

**PrimitiveTopology::Code** **GetPrimitiveTopology** () const
- get the primitive topology

void **SetBoundingBox** (const Math::bbox &b)
- set the primitive group's local bounding box

const Math::bbox & **GetBoundingBox** () const
- get the primitive group's local bounding box

SizeT **GetNumPrimitives** () const
- get computed number of primitives
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><code>enumeration</code></td>
</tr>
</tbody>
</table>
## Static Public Member Functions

<table>
<thead>
<tr>
<th>static Code</th>
<th><code>FromString</code> (const <code>Util::String</code> &amp;str)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>convert from string</td>
</tr>
<tr>
<td>static <code>Util::String</code></td>
<td><code>ToString</code> (Code code)</td>
</tr>
<tr>
<td></td>
<td>convert to string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Tue Feb 19 12:16:44 2008
CoreGraphics::RenderDevice
#include <renderdevice.h>

Inheritance diagram for 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>RenderDevice ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~RenderDevice ()</td>
<td>destructor</td>
</tr>
<tr>
<td>IDirect3DDevice9 * GetDirect3DDevice () const</td>
<td>get pointer to the d3d device</td>
</tr>
<tr>
<td>bool Open ()</td>
<td>open the device</td>
</tr>
<tr>
<td>void Close ()</td>
<td>close the device</td>
</tr>
<tr>
<td>bool BeginFrame ()</td>
<td>begin complete frame</td>
</tr>
<tr>
<td>void SetVertexBuffer(const Ptr<a href="">CoreGraphics::VertexBuffer</a> &amp;vb)</td>
<td>set current vertex buffer</td>
</tr>
<tr>
<td>void SetIndexBuffer(const Ptr<a href="">CoreGraphics::IndexBuffer</a> &amp;ib)</td>
<td>set current index buffer</td>
</tr>
<tr>
<td>void Draw ()</td>
<td>draw current primitives</td>
</tr>
<tr>
<td>void EndPass ()</td>
<td>end current pass</td>
</tr>
<tr>
<td>void EndFrame ()</td>
<td>end complete frame</td>
</tr>
<tr>
<td>void Present ()</td>
<td>present the rendered scene</td>
</tr>
<tr>
<td>void SaveScreenshot(const CoreGraphics::ImageFileFormat::Code fmt, const Ptr<a href="">IO::Stream</a> &amp;outStream)</td>
<td>save a screenshot to the provided stream</td>
</tr>
<tr>
<td>bool IsOpen () const</td>
<td>return true if currently open</td>
</tr>
<tr>
<td>AttachEventHandler (const Ptr&lt;</td>
<td></td>
</tr>
</tbody>
</table>
void CoreGraphics::RenderEventHandler (CoreGraphics::RenderEventHandler &h)
attach a render event handler

void RemoveEventHandler (CoreGraphics::RenderEventHandler &h)
remove a render event handler

const Ptr < CoreGraphics::RenderTarget > & GetDefaultRenderTarget () const
get default render target

void BeginPass (CoreGraphics::RenderTarget &rt,
const Ptr < CoreGraphics::ShaderInstance > &passShader)
bEGIN rendering a frame pass

void BeginBatch (CoreGraphics::BatchType::Code batchType,
const Ptr < CoreGraphics::ShaderInstance > &batchShader)
bEGIN rendering a batch inside

const Ptr < CoreGraphics::VertexBuffer > & GetVertexBuffer () const
get current vertex buffer

const Ptr < CoreGraphics::IndexBuffer > & GetIndexBuffer () const
get current index buffer

void SetPrimitiveGroup (const CoreGraphics::PrimitiveGroup &pg)
set current primitive group

const CoreGraphics::PrimitiveGroup & GetPrimitiveGroup () const
get current primitive group

void EndBatch ()
end current batch

bool IsInBeginFrame () const
check if inside BeginFrame

int GetRefCount () const

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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</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 IDirect3D9 * <strong>GetDirect3D</strong> ()</td>
<td>get pointer to Direct3D interface, opens Direct3D if not happened yet</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>
Protected Member Functions

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 render device. The RenderEvent::DeviceClose will be sent to all registered event handlers.

Reimplemented from **Base::RenderDeviceBase**.

```c++
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**.

```c++
void Direct3D9::D3D9RenderDevice::SetVertexBuffer(const Ptr<CoreGraphics::VertexBuffer> &vb) [inherited]
```

**set current vertex buffer**

Sets the vertex buffer to use for the next **Draw()**.

Reimplemented from **Base::RenderDeviceBase**.

```c++
void Direct3D9::D3D9RenderDevice::SetIndexBuffer(const Ptr<CoreGraphics::IndexBuffer> &ib) [inherited]
```

**set current index buffer**

Sets the index buffer to use for the next **Draw()**.

Reimplemented from **Base::RenderDeviceBase**.

```c++
void Direct3D9::D3D9RenderDevice::Draw() [inherited]
```
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::EndPass()
```

end current pass

End the current rendering pass. This will flush all texture stages in order to keep the d3d9 resource reference consistent without too much hassle.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
void Direct3D9::D3D9RenderDevice::EndFrame()
```

end complete frame

End a complete frame. Call this once per frame after rendering and presentation has happened, and only if **BeginFrame()** returns true.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
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**.

```cpp
void Direct3D9::D3D9RenderDevice::SaveScreenshot(CoreGraphics::ImageFileFormat::Code fmt,
```
save a screenshot to the provided stream
Save the backbuffer to the provided stream.
Reimplemented from **Base::RenderDeviceBase**.

```cpp
void Base::RenderDeviceBase::AttachEventHandler(const Ptr<CoreGraphics::RenderEventHandler> & h) [inherited]
```

**attach a render event handler**
Attach an event handler to the render device.

```cpp
void Base::RenderDeviceBase::RemoveEventHandler(const Ptr<CoreGraphics::RenderEventHandler> & h) [inherited]
```

**remove a render event handler**
Remove an event handler from the display device.

```cpp
bool Base::RenderDeviceBase::NotifyEventHandlers(const CoreGraphics::RenderEvent & 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::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><strong>Code</strong></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><strong>RenderEvent ()</strong></td>
<td>Default constructor</td>
</tr>
<tr>
<td><strong>RenderEvent (Code c)</strong></td>
<td>Constructor with event code</td>
</tr>
<tr>
<td><strong>GetEventCode () const</strong></td>
<td>Get event code</td>
</tr>
</tbody>
</table>
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><code>~RenderEventHandler ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual void OnAttach ()</code></td>
<td>called when the event handler is attached to the <code>RenderDevice</code></td>
</tr>
<tr>
<td><code>virtual void OnRemove ()</code></td>
<td>called when the event handler is removed from the <code>RenderDevice</code></td>
</tr>
<tr>
<td><code>virtual bool PutEvent (const RenderEvent &amp;event)</code></td>
<td>called by <code>RenderDevice</code> when an event happens</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) 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>
</tbody>
</table>
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>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>virtual bool</th>
<th><strong>HandleEvent</strong> (const <strong>RenderEvent</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::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.

```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::RenderTarget**
CoreGraphics::RenderTarget Class Reference

#include <rendertarget.h>

Inheritance diagram for CoreGraphics::RenderTarget:

```
Core::RefCounted
  ↓
Base::RenderTargetBase
  ↓
Direct3D9::D3D9RenderTarget
  ↓
CoreGraphics::RenderTarget
```
Detailed Description

A render target 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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void Setup ()</code></td>
<td>setup the render target object</td>
</tr>
<tr>
<td><code>void Discard ()</code></td>
<td>discard the render target object</td>
</tr>
<tr>
<td><code>void BeginPass ()</code></td>
<td>begin a render pass</td>
</tr>
<tr>
<td><code>void EndPass ()</code></td>
<td>end current render pass</td>
</tr>
<tr>
<td><code>void GenerateMipLevels ()</code></td>
<td>generate mipmap levels</td>
</tr>
<tr>
<td><code>bool IsDefaultRenderTarget () const</code></td>
<td>get default render target flag</td>
</tr>
<tr>
<td><code>bool HasColorBuffer (IndexT colorBufferIndex) const</code></td>
<td>return true if color buffer exists</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>bool IsValid () const</code></td>
<td>return true if valid (has been setup)</td>
</tr>
<tr>
<td><code>void SetWidth (SizeT w)</code></td>
<td>set render target width</td>
</tr>
<tr>
<td><code>SizeTGetWidth () 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>
</tbody>
</table>
### Methods

- **void** `AddColorBuffer(CoreGraphics::PixelFormat::Code colorFormat)`
  
  *add a color buffer*

- **SizeT** `GetNumColorBuffers()` const
  
  *get number of color buffers*

- **CoreGraphics::PixelFormat::Code** `GetColorBufferFormat(IndexT colorBufferIndex)` const
  
  *get color buffer format at index*

- **void** `AddDepthStencilBuffer()`
  
  *add a depth/stencil buffer*

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

- **bool** `AreMipMapsEnabled()` const
  
  *get mipmap generation flag*

- **void** `SetResolveTextureResourceResourceld(const Resources::ResourceId &resId)`
  
  *set resolve texture resource id*

- **const Resources::ResourceId &** `GetResolveTextureResourceResourceld()` const
  
  *get resolve 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** `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< uint > &r)
set the current resolve rectangle (in pixels)

const Math::rectangle< uint > & 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

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><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

static void DumpRefCountingLeaks ()
  dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
**Static Public Attributes**

<table>
<thead>
<tr>
<th>static const IndexT</th>
<th>MaxNumColorBuffers</th>
<th>=</th>
<th>4</th>
</tr>
</thead>
</table>

*max number of color buffers*
Protected Member Functions

void SetupMultiSampleType ()
setup compatible multisample type

void SetDefaultRenderTarget (bool b)
set to true if default render target
Member Function Documentation

```cpp
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.

```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::Shader
#include <shader.h>

Inheritance diagram for CoreGraphics::Shader:

```
          Core::RefCounted
            ↑
      Resources::Resource
        ↑
    Base::ShaderBase
      ↑
Direct3D9::D3D9Shader
      ↑
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></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>virtual void Unload ()</td>
<td>Unload the resource or cancel the pending load</td>
</tr>
<tr>
<td>ID3DXEffect * GetD3D9Effect () const</td>
<td>Get pointer to D3D effect</td>
</tr>
<tr>
<td>&lt; CoreGraphics::ShaderInstance &gt; CreateShaderInstance ()</td>
<td>Create a shader instance from this shader</td>
</tr>
<tr>
<td>void DiscardShaderInstance (const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp;inst)</td>
<td>Discard a shader instance</td>
</tr>
<tr>
<td>const Util::Array&lt; Ptr&lt; CoreGraphics::ShaderInstance &gt; &gt; &amp; GetAllShaderInstances () const</td>
<td>Get all instances</td>
</tr>
<tr>
<td>void SetAsyncEnabled (bool b)</td>
<td>Request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td>bool IsAsyncEnabled () const</td>
<td>Return true if asynchronous resource loading is enabled</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>
</tbody>
</table>
const Ptr & ResourceSaver> & GetSaver() const

get optional resource saver

SizeT GetUseCount() const

get current use count

virtual State Load()  
load the resource

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
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 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>void SetState(State S)</td>
<td>set current state</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>
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::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>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>typedef unsigned int</code></td>
<td><strong>Mask</strong></td>
<td>a shader feature bit mask</td>
</tr>
<tr>
<td><code>typedef Util::Atom&lt; Util::String &gt;</code></td>
<td><strong>Name</strong></td>
<td>a single shader feature name</td>
</tr>
</tbody>
</table>
Public Member Functions

ShaderFeature ()

constructor
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
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ID3DXEffect * GetD3D9Effect () const</code></td>
<td>get pointer to d3d9 effect object</td>
</tr>
<tr>
<td><code>bool SelectActiveVariation(CoreGraphics::ShaderFeature::Mask featureMask)</code></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 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 instance no longer needed</td>
</tr>
<tr>
<td><code>bool IsValid () const</code></td>
<td>return true if this object is valid</td>
</tr>
<tr>
<td><code>const Ptr &lt; CoreGraphics::Shader &gt; &amp; GetOriginalShader () const</code></td>
<td>get pointer to original shader which created this instance</td>
</tr>
<tr>
<td><code>bool HasVariableByName (const CoreGraphics::ShaderVariable::Name &amp;n) const</code></td>
<td>return true if the shader instance has a variable by name</td>
</tr>
<tr>
<td><code>bool HasVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &amp;n) const</code></td>
<td>return true if shader has variable by semantic</td>
</tr>
<tr>
<td><code>SizeT GetNumVariables () const</code></td>
<td>get number of variables</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetVariableByIndex(IndexT i)</code></td>
<td>get a variable by index</td>
</tr>
<tr>
<td><code>GetVariableByName(const CoreGraphics::ShaderVariable::Name &amp;n)</code></td>
<td>get a variable by name</td>
</tr>
<tr>
<td><code>GetVariableBySemantic(const CoreGraphics::ShaderVariable::Semantic &amp;s)</code></td>
<td>get a variable by semantic</td>
</tr>
<tr>
<td><code>HasVariation(CoreGraphics::ShaderFeature::Mask featureMask)</code></td>
<td>return true if variation exists by matching feature mask</td>
</tr>
<tr>
<td><code>GetNumVariations()</code></td>
<td>get number of variations in the shader</td>
</tr>
<tr>
<td><code>GetVariationByIndex(IndexT i)</code></td>
<td>get shader variation by index</td>
</tr>
<tr>
<td><code>GetVariationByFeatureMask(CoreGraphics::ShaderFeature::Mask featureMask)</code></td>
<td>get shader variation by feature mask</td>
</tr>
<tr>
<td><code>GetActiveVariation()</code></td>
<td>get currently active variation</td>
</tr>
<tr>
<td><code>AddPreShader(const CoreGraphics::PreShader &amp;preShader)</code></td>
<td>add a pre-shader</td>
</tr>
<tr>
<td><code>RemovePreShader(const CoreGraphics::PreShader &amp;preShader)</code></td>
<td>remove a pre-shader</td>
</tr>
<tr>
<td><code>GetPreShaders()</code></td>
<td>get array of pre-shaders</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><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

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>virtual void <strong>Setup</strong> (const Ptr&lt; CoreGraphics::Shader &gt; &amp;origShader)</td>
<td>setup the shader instance from its original shader object</td>
</tr>
<tr>
<td>virtual void <strong>Cleanup</strong> ()</td>
<td>cleanup the shader instance</td>
</tr>
<tr>
<td>void <strong>OnLostDevice</strong> ()</td>
<td>called by d3d9 shader server when d3d9 device is lost</td>
</tr>
<tr>
<td>void <strong>OnResetDevice</strong> ()</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, inherited]]

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]

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::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 Types

<table>
<thead>
<tr>
<th>enum</th>
<th>ShaderParamBindMode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shader parameter bind modes</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ShaderServer ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~ShaderServer ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>Open ()</code></td>
<td>open the shader server</td>
</tr>
<tr>
<td><code>Close ()</code></td>
<td>close the shader server</td>
</tr>
<tr>
<td><code>bool HasSharedVariableByName (const CoreGraphics::ShaderVariable::Name &amp;name) const</code></td>
<td>return true if a shared variable exists by name</td>
</tr>
<tr>
<td><code>bool HasSharedVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &amp;sem) const</code></td>
<td>return true if a shared variable exists by semantic</td>
</tr>
<tr>
<td><code>SizeT GetNumSharedVariables () const</code></td>
<td>get number of shared variables</td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::ShaderVariable&gt; &amp; GetSharedVariableByIndex (IndexT i)</code></td>
<td>get a shared variable by index</td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::ShaderVariable&gt; &amp; GetSharedVariableByName (const CoreGraphics::ShaderVariable::Name &amp;name)</code></td>
<td>get a shared variable by name</td>
</tr>
<tr>
<td><code>constPtr&lt;CoreGraphics::ShaderVariable&gt; &amp; GetSharedVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &amp;sem)</code></td>
<td>get a shared variable by semantic</td>
</tr>
<tr>
<td><code>ID3DXEffectPool * GetD3D9EffectPool () const</code></td>
<td>get pointer to global effect pool</td>
</tr>
<tr>
<td><code>void SetShaderParamBindMode (ShaderParamBindMode m)</code></td>
<td>set shader param bind mode (by name or by semantic,</td>
</tr>
<tr>
<td><strong>ShaderParamBindMode</strong></td>
<td><strong>GetShaderParamBindMode() const</strong></td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td></td>
<td>get shader param bind mode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>bool</strong></th>
<th><strong>IsOpen() const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if the shader server is open</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>bool</strong></th>
<th><strong>HasShader(const Resources::ResourceId &amp;resId) const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return true if a shader exists</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Ptr&lt; CoreGraphics::ShaderInstance</strong></th>
<th><strong>CreateShaderInstance(const Resources::ResourceId &amp;resId)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>create a new shader instance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>const Util::Dictionary&lt; Resources::ResourceId, Ptr&lt; CoreGraphics::Shader &gt; &gt; &amp;</strong></th>
<th><strong>GetAllShaders() const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get all loaded shaders</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>void</strong></th>
<th><strong>SetActiveShaderInstance(const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp;shaderInst)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set currently active shader instance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp;</strong></th>
<th><strong>GetActiveShaderInstance() const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get currently active shader instance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>void</strong></th>
<th><strong>ResetFeatureBits()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>reset the current feature bits</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>void</strong></th>
<th><strong>SetFeatureBits(CoreGraphics::ShaderFeature::Mask m)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set shader feature by bit mask</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>void</strong></th>
<th><strong>ClearFeatureBits(CoreGraphics::ShaderFeature::Mask m)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>clear shader feature by bit mask</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CoreGraphics::ShaderFeature::Mask</strong></th>
<th><strong>GetFeatureBits() const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>get the current feature mask</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CoreGraphics::ShaderFeature::Mask</strong></th>
<th><strong>FeatureStringToMask(const Util::String &amp;str)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>convert a shader feature string into a feature bit mask</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::String</strong></th>
<th><strong>FeatureMaskToString(CoreGraphics::ShaderFeature::Mask m)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Function/Method</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>
Member Function Documentation

`Ptr< ShaderInstance >`  
Base::ShaderServerBase::CreateShaderInstance (  
  `const 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`
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::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>enum</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shader variable types</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>typedef Util::Atom<a href="">Util::String</a></th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shader variable name typedef</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>typedef Util::Atom<a href="">Util::String</a></th>
<th>Semantic</th>
</tr>
</thead>
<tbody>
<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>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>SetInt</td>
<td>int value</td>
<td>set int value</td>
</tr>
<tr>
<td>void</td>
<td>SetIntArray</td>
<td>const int *values, SizeT count</td>
<td>set int array values</td>
</tr>
<tr>
<td>void</td>
<td>SetFloat</td>
<td>float value</td>
<td>set float value</td>
</tr>
<tr>
<td>void</td>
<td>SetFloatArray</td>
<td>const float *values, SizeT count</td>
<td>set float array values</td>
</tr>
<tr>
<td>void</td>
<td>SetVector</td>
<td>const Math::float4 &amp;value</td>
<td>set vector value</td>
</tr>
<tr>
<td>void</td>
<td>SetVectorArray</td>
<td>const Math::float4 *values, SizeT count</td>
<td>set vector array values</td>
</tr>
<tr>
<td>void</td>
<td>SetMatrix</td>
<td>const Math::matrix44 &amp;value</td>
<td>set matrix value</td>
</tr>
<tr>
<td>void</td>
<td>SetMatrixArray</td>
<td>const Math::matrix44 *values, SizeT count</td>
<td>set matrix array values</td>
</tr>
<tr>
<td>void</td>
<td>SetBool</td>
<td>bool value</td>
<td>set bool value</td>
</tr>
<tr>
<td>void</td>
<td>SetBoolArray</td>
<td>const bool *values, SizeT count</td>
<td>set bool array values</td>
</tr>
<tr>
<td>void</td>
<td>SetTexture</td>
<td>const Ptr<a href="">CoreGraphics::Texture</a> &amp;value</td>
<td>set texture value</td>
</tr>
<tr>
<td>Ptr</td>
<td>CreateInstance</td>
<td>()</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------</td>
<td>------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>create a shader variable instance</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Type</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>GetType () const</code></td>
<td>get the data type of the variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>const Name &amp;</strong> <code>GetName () const</code></td>
<td>get the name of the variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>const Semantic &amp;</strong> <code>GetSemantic () const</code></td>
<td>get the semantics of the variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>GetNumArrayElements () const</code></td>
<td>number of array elements if this is an array variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>bool</strong> <code>IsArray () const</code></td>
<td>return true if this is an array</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>int</strong> <code>GetRefCount () const</code></td>
<td>get the current refcount</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>void</strong> <code>AddRef ()</code></td>
<td>increment refcount by one</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>void</strong> <code>Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>bool</strong> <code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>bool</strong> <code>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></td>
<td><strong>bool</strong> <code>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></td>
<td><strong>bool</strong> <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>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>bool</strong> <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</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<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 string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td>get the class FourCC code</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>static Util::String</code></td>
<td><strong>TypeToString</strong> <em>(Type t)</em> <strong>convert type to string</strong></td>
</tr>
<tr>
<td><code>static void</code></td>
<td><strong>DumpRefCountingLeaks</strong> <em>(void)</em> *<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 SetType (Type t)</code></td>
<td>set variable type</td>
</tr>
<tr>
<td><code>void SetName (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>
<tr>
<td><code>void SetNumArrayElements (SizeT n)</code></td>
<td>set number of array elements</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]

deprecated

increment refcount by one

Increment the refcount of the object.

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

dump refcounting leaks, call at end of application (NEBULA3_DEBUG

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

generateString

get the class name

Get the class name of the object.

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

generateString

generateFourCC

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>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const Ptr&lt;CoreGraphics::ShaderVariable&gt; &amp; GetShaderVariable()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>void Apply()</code></td>
<td>void</td>
</tr>
<tr>
<td><code>void SetInt (int value)</code></td>
<td>void</td>
</tr>
<tr>
<td><code>void SetIntArray (const int *values, SizeT count)</code></td>
<td>void</td>
</tr>
<tr>
<td><code>void SetFloat (float value)</code></td>
<td>void</td>
</tr>
<tr>
<td><code>void SetFloatArray (const float *values, SizeT count)</code></td>
<td>void</td>
</tr>
<tr>
<td><code>void SetVector (const Math::float4 &amp;value)</code></td>
<td>void</td>
</tr>
<tr>
<td><code>void SetVectorArray (const Math::float4 *values, SizeT count)</code></td>
<td>void</td>
</tr>
<tr>
<td><code>void SetMatrix (const Math::matrix44 &amp;value)</code></td>
<td>void</td>
</tr>
<tr>
<td><code>void SetMatrixArray (const Math::matrix44 *values, SizeT count)</code></td>
<td>void</td>
</tr>
<tr>
<td><code>void SetBool (bool value)</code></td>
<td>void</td>
</tr>
<tr>
<td><code>void SetBoolArray (const bool *values, SizeT count)</code></td>
<td>void</td>
</tr>
<tr>
<td><code>void SetTexture (const Ptr&lt;CoreGraphics::Texture&gt; &amp;value)</code></td>
<td>void</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, 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
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]

going 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

**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>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3DXHANDLE</td>
<td><code>GetD3D9Technique()</code> const</td>
<td>get the D3DX technique handle</td>
</tr>
<tr>
<td>ID3DXEffect *</td>
<td><code>GetD3D9Effect()</code> const</td>
<td>get the D3DX effect which owns this variation</td>
</tr>
<tr>
<td>const Name &amp;</td>
<td><code>GetName()</code> const</td>
<td>get the shader variation’s name</td>
</tr>
<tr>
<td>CoreGraphics::ShaderFeature::Mask</td>
<td><code>GetFeatureMask()</code> const</td>
<td>get the feature bit mask of this variation</td>
</tr>
<tr>
<td>SizeT</td>
<td><code>GetNumPasses()</code> const</td>
<td>get number of passes in this variation</td>
</tr>
<tr>
<td>int</td>
<td><code>GetRefCount()</code> const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void</td>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td><code>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><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</td>
<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>bool</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>
</tr>
<tr>
<td>bool</td>
<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>bool</td>
<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>bool</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td>------</td>
<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>
<td></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
<td></td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
<td></td>
</tr>
<tr>
<td>get the class FourCC code</td>
<td></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><code>SetName</code> (const <code>Name</code> &amp;n)</td>
<td>set variation name</td>
</tr>
<tr>
<td><code>SetFeatureMask</code> (CoreGraphics::ShaderFeature::Mask m)</td>
<td>set feature bit mask of this variation</td>
</tr>
<tr>
<td><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.
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 Types

<table>
<thead>
<tr>
<th>enum</th>
<th>ShapeType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shape type</td>
</tr>
</tbody>
</table>
Public Member Functions

**ShapeRenderer ()**
*constructor*

**virtual ~ShapeRenderer ()**
*destructor*

**void Open ()**
*open the shape renderer*

**void Close ()**
*close the shape renderer*

**void DrawShape (const Math::matrix44 &modelTransform, ShapeType shapeType, const Math::float4 &color)**
*draw a shape*

**void DrawPrimitives (const Math::matrix44 &modelTransform, CoreGraphics::PrimitiveTopology::Code topology, SizeT numPrimitives, float *vertices, SizeT vertexWidth, const Math::float4 &color)**
*draw primitives*

**void DrawIndexedPrimitives (const Math::matrix44 &modelTransform, CoreGraphics::PrimitiveTopology::Code topology, SizeT numPrimitives, float *vertices, SizeT numVertices, SizeT vertexWidth, void *indices, CoreGraphics::IndexType::Code indexType, const Math::float4 &color)**
*draw indexed primitives*

**bool isOpen () const**
$return true if open*

**void Begin ()**
*begin drawing shapes*

**void End ()**
*end drawing shapes*

**int GetRefCount () const**
*get the current refcount*
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
<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

```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
builds only!

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

#include <streamanimationloader.h>

Inheritance diagram for CoreGraphics::StreamAnimationLoader:
Detailed Description

Setup a animation object from a stream. Supports the following file formats:

- nax2 (Nebula2 binary animation file format)
- nax3 (Nebula3 binary animation file format)
- nam3 (Nebula3 ascii animation 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>StreamAnimationLoader ()</strong></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td>virtual ~StreamAnimationLoader ()</td>
<td><em>destructor</em></td>
</tr>
<tr>
<td>virtual bool CanLoadAsync () const</td>
<td><em>return true if asynchronous loading is supported</em></td>
</tr>
<tr>
<td>virtual bool OnLoadRequested ()</td>
<td><em>called by resource when a load is requested</em></td>
</tr>
<tr>
<td>virtual void OnLoadCancelled ()</td>
<td><em>called by resource to cancel a pending load</em></td>
</tr>
<tr>
<td>virtual bool OnPending ()</td>
<td><em>call frequently while after OnLoadRequested() to put Resource into loaded state</em></td>
</tr>
<tr>
<td>virtual void OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res)</td>
<td><em>called when the resource loader is attached to its resource</em></td>
</tr>
<tr>
<td>virtual void OnRemoveFromResource ()</td>
<td><em>called when the resource loader is removed from its resource</em></td>
</tr>
<tr>
<td>bool IsAttachedToResource () const</td>
<td><em>return true if attached to resource</em></td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetResource () const</td>
<td><em>get pointer to resource</em></td>
</tr>
<tr>
<td>Resource::State GetState () const</td>
<td><em>return current state</em></td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td><em>get the current refcount</em></td>
</tr>
<tr>
<td>void AddRef ()</td>
<td><em>increment refcount by one</em></td>
</tr>
<tr>
<td>void Release ()</td>
<td><em>decrement refcount and destroy object if refcount is zero</em></td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td><em>return true if this object is instance of given class</em></td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className)</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>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) 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

```cpp
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.
CoreGraphics::StreamMeshLoader
CoreGraphics::StreamMeshLoader
Class Reference

#include <streammeshloader.h>

Inheritance diagram for CoreGraphics::StreamMeshLoader:
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><code>StreamMeshLoader()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual destructor</td>
<td><code>~StreamMeshLoader()</code></td>
</tr>
<tr>
<td>virtual bool <code>CanLoadAsync()</code> const</td>
<td>return true if asynchronous loading is supported</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 Resource 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>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>
</tbody>
</table>
| bool `IsInstanceOf(const Util::String &className)` | }
<table>
<thead>
<tr>
<th>const</th>
<th>return true if this object is instance of given class by string</th>
</tr>
</thead>
<tbody>
<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

```
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
**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
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

Direct3D::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>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>OnAttachToResource</strong> (const <strong>Ptr&lt; Resource &gt;</strong> &amp;res)</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;</strong> &amp; <strong>GetResource</strong> () const</td>
<td>get pointer to resource</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>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 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>-----------------------------------</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><code>DumpRefCountingLeaks</code> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------</td>
</tr>
</tbody>
</table>

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

void **SetState** (Resource::State S)

*set current state*
Member Function Documentation

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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>virtual bool CanLoadAsync () const return true if asynchronous loading is supported</td>
</tr>
<tr>
<td>virtual bool OnLoadRequested ()</td>
<td>virtual bool OnLoadRequested () called by resource when a load is requested</td>
</tr>
<tr>
<td>virtual void OnLoadCancelled ()</td>
<td>virtual void OnLoadCancelled () called by resource to cancel a pending load</td>
</tr>
<tr>
<td>virtual bool OnPending ()</td>
<td>virtual bool OnPending () called 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>virtual void OnAttachToResource (const Ptr&lt; Resource &gt; &amp;res) called when the resource loader is attached to its resource</td>
</tr>
<tr>
<td>virtual void OnRemoveFromResource ()</td>
<td>virtual void OnRemoveFromResource () called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td>bool IsAttachedToResource () const</td>
<td>bool IsAttachedToResource () const return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt; Resource &gt; &amp; GetResource () const</td>
<td>const Ptr&lt; Resource &gt; &amp; GetResource () const get pointer to resource</td>
</tr>
<tr>
<td>Resource::State GetState () const</td>
<td>Resource::State GetState () const return current state</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>int GetRefCount () const get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td>void AddRef () increment refcount by one</td>
</tr>
<tr>
<td>void Release ()</td>
<td>void Release () decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td>bool IsInstanceOf (const Util::String &amp;className) const 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>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const 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></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!)
Protected Member Functions

```c
void SetState (Resource::State S)  
set current state
```
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.
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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>virtual bool</strong> OnSave ()</td>
<td>called by resource when a save is requested</td>
</tr>
<tr>
<td>void SetStream (const Ptr<a href="">IO::Stream</a> &amp;stream)</td>
<td>set stream to save to</td>
</tr>
<tr>
<td>const Ptr<a href="">IO::Stream</a> &amp; GetStream () const</td>
<td>get save-stream</td>
</tr>
<tr>
<td>void SetFormat (CoreGraphics::ImageFileFormat::Code fmt)</td>
<td>set file format (default is JPG)</td>
</tr>
<tr>
<td>CoreGraphics::ImageFileFormat::Code GetFormat () const</td>
<td>get file format</td>
</tr>
<tr>
<td>void SetMipLevel (IndexT mipLevel)</td>
<td>set the mip level to save (default is 0, the top level)</td>
</tr>
<tr>
<td>IndexT GetMipLevel () const</td>
<td>get the mip level to save</td>
</tr>
<tr>
<td><strong>virtual void</strong> OnAttachToResource (const Ptr&lt;Resource&gt; &amp;res)</td>
<td>called when the resource saver is attached to its resource</td>
</tr>
<tr>
<td><strong>virtual void</strong> OnRemoveFromResource ()</td>
<td>called when the resource saver 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>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</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (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>bool</td>
<td>IsInstanceOf (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>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 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.
| CoreGraphics::Texture |
CoreGraphics::Texture Class Reference

#include <texture.h>

Inheritance diagram for CoreGraphics::Texture:

```
Core::RefCounted
    |
    v
Resources::Resource
    |
    v
Base::ResourceBase
    |
    v
Base::TextureBase
    |
    v
Direct3D9::D3D9Texture
    |
    v
CoreGraphics::Texture
```
Detailed Description

Front-end class for texture objects.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Texture types</td>
</tr>
<tr>
<td>CubeFace</td>
<td>Cube map face</td>
</tr>
<tr>
<td>Usage</td>
<td>Resource usage flags</td>
</tr>
<tr>
<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>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>bool Map (IndexT mipLevel, Access accessMode, MapInfo &amp;outMapInfo)</td>
<td>Map the 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>bool MapCubeFace (CubeFace face, IndexT mipLevel, Access accessMode, 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>Type GetType () const</td>
<td>Get texture type</td>
</tr>
<tr>
<td>SizeT GetWidth () const</td>
<td>Get width of texture</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>getSizeT()</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>CoreGraphics::PixelFormat::getCode()</code></td>
<td>get pixel format of the texture</td>
</tr>
<tr>
<td><code>GetUsage()</code></td>
<td>get resource usage 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 loading</td>
</tr>
<tr>
<td><code>IsAsyncEnabled()</code></td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>SetResourceId(const ResourceId &amp;id)</code></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><code>GetResourceId()</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()</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()</code></td>
<td>get optional resource saver</td>
</tr>
<tr>
<td><code>GetUseCount()</code></td>
<td>get current use count</td>
</tr>
<tr>
<td><code>virtual State Load()</code></td>
<td></td>
</tr>
</tbody>
</table>
### load the resource

**State**
- **GetState**() const
  - get current state

**bool**
- **IsLoaded**() const
  - return true if current state is Loaded
- **IsPending**() const
  - return true if current state is Pending
- **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
- **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 &className]) const
  - return true if this object is instance of given class by string
- **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
- **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 &rttiFourCC]) const
  - 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><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<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>void SetupFromD3D9Texture (IDirect3DTexture9 *ptr)</code></td>
<td>setup from a IDirect3DTexture9</td>
</tr>
<tr>
<td><code>void SetupFromD3D9CubeTexture (IDirect3DCubeTexture9 *ptr)</code></td>
<td>setup from a IDirect3DCubeTexture</td>
</tr>
<tr>
<td><code>void SetupFromD3D9VolumeTexture (IDirect3DVolumeTexture9 *ptr)</code></td>
<td>setup from a IDirect3DVolumeTexture</td>
</tr>
<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 SetUsage (Usage usage)</code></td>
<td>set 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>void setState (State s)</code></td>
<td>set current state</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(IDirect3DTexture9* tex2D) [protected, inherited]

setup from a IDirect3DTexture9

Helper method to setup the texture object from a D3D9 2D texture.

void Direct3D9::D3D9Texture::SetupFromD3D9CubeTexture(IDirect3DCubeTexture9* texCube) [protected, inherited]

setup from a IDirect3DCubeTexture

Helper method to setup the texture object from a D3D9 cube texture.

void Direct3D9::D3D9Texture::SetupFromD3D9VolumeTexture(IDirect3DVolumeTexture9* texVolume)

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.

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::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 than 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/Member</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</strong> (const <strong>DisplayEvent</strong> &amp;event)</td>
<td>called by <strong>DisplayDevice</strong> 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 <strong>DisplayDevice</strong></td>
</tr>
<tr>
<td>virtual void <strong>OnRemove</strong> ()</td>
<td>called when the event handler is removed from the <strong>DisplayDevice</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>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>
</tbody>
</table>
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>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`.
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

Inheritance diagram for
CoreGraphics::ThreadSafeRenderEventHandler:

```
#include <threadsaferendereventhandler.h>

CoreGraphics::ThreadSafeRenderEventHandler
```

Class Reference
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/Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ThreadSafeRenderEventHandler</strong> ()</td>
</tr>
<tr>
<td><em>constructor</em></td>
</tr>
<tr>
<td><strong>virtual ~ThreadSafeRenderEventHandler</strong> ()</td>
</tr>
<tr>
<td><em>destructor</em></td>
</tr>
<tr>
<td><strong>virtual bool PutEvent</strong> (const RenderEvent &amp;event)</td>
</tr>
<tr>
<td><em>called by RenderDevice when an event happens</em></td>
</tr>
<tr>
<td><strong>void HandlePendingEvents</strong> ()</td>
</tr>
<tr>
<td><em>handle all pending events (called by consumer thread)</em></td>
</tr>
<tr>
<td><strong>virtual void OnAttach</strong> ()</td>
</tr>
<tr>
<td><em>called when the event handler is attached to the RenderDevice</em></td>
</tr>
<tr>
<td><strong>virtual void OnRemove</strong> ()</td>
</tr>
<tr>
<td><em>called when the event handler is removed from the RenderDevice</em></td>
</tr>
<tr>
<td><strong>int GetRefCount</strong> () const</td>
</tr>
<tr>
<td><em>get the current refcount</em></td>
</tr>
<tr>
<td><strong>void AddRef</strong> ()</td>
</tr>
<tr>
<td><em>increment refcount by one</em></td>
</tr>
<tr>
<td><strong>void Release</strong> ()</td>
</tr>
<tr>
<td><em>decrement refcount and destroy object if refcount is zero</em></td>
</tr>
<tr>
<td><strong>bool 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><strong>bool 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><strong>bool 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><strong>bool 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><strong>bool 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>bool IsA</strong> (const Util::FourCC &amp;rttiFourCC) const</td>
</tr>
<tr>
<td><em>return true if this object is instance of given class, or a derived class, by fourcc</em></td>
</tr>
<tr>
<td>const <code>Util::String &amp;</code></td>
</tr>
<tr>
<td>-----------------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><code>Util::FourCC</code></td>
</tr>
<tr>
<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>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

```cpp
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.
```

```cpp
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.
```

```cpp
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.
```

```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::TransformDevice
CoreGraphics::TransformDevice Class Reference

#include <transformdevice.h>
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>Method</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>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](http://doxygen.org) at Tue Feb 19 12:16:45 2008
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><strong>Usage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>resource usage flags</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>State</strong></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><code>virtual void Unload ()</code></td>
<td>Unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td><code>void * Map (MapType mapType)</code></td>
<td>Map the vertices for CPU access</td>
</tr>
<tr>
<td><code>void Unmap ()</code></td>
<td>Unmap the resource</td>
</tr>
<tr>
<td><code>IDirect3DVertexBuffer9 * GetD3D9VertexBuffer () const</code></td>
<td>Get pointer to d3d9 vertex buffer object</td>
</tr>
<tr>
<td><code>const Ptr &lt; CoreGraphics::VertexLayout &gt; &amp; GetVertexLayout () const</code></td>
<td>Get the vertex layout</td>
</tr>
<tr>
<td><code>SizeT GetNumVertices () const</code></td>
<td>Get number of vertices in the buffer</td>
</tr>
<tr>
<td><code>Usage GetUsage () const</code></td>
<td>Get resource usage 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/async 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 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</code></td>
<td>Set the resource saver</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>const Ptr &amp; ResourceSaver 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>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>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</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>SetVertexLayout</code></td>
<td>(const <code>Ptr&lt; CoreGraphics::VertexLayout &gt;</code> &amp;vertexLayout) set vertex layout (set by resource loader)</td>
</tr>
<tr>
<td><code>SetNumVertices</code></td>
<td>(SizeT numVertices) set number of vertices (set by resource loader)</td>
</tr>
<tr>
<td><code>SetUsage</code></td>
<td>(Usage usage) set resource usage type</td>
</tr>
<tr>
<td><code>SetAccess</code></td>
<td>(Access access) set resource cpu access type</td>
</tr>
<tr>
<td><code>SetState</code></td>
<td>(State s) set current state</td>
</tr>
<tr>
<td><code>IncrUseCount</code></td>
<td>() increment use count</td>
</tr>
<tr>
<td><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.

**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.

\texttt{const \textit{Util::String} \& Core::RefCounted::GetClassName() const [inline, inherited]}

get the class name

Get the class name of the object.

\texttt{Util::FourCC Core::RefCounted::GetClassFourCC() const [inline, inherited]}

get the class FourCC code

Get the class FourCC of the object.

\texttt{void Core::RefCounted::DumpRefCountingLeaks() [static, inherited]}

dump 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 Types

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>SemanticName</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>component semantic</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>Format</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>component format</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>AccessType</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>access type</em></td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>VertexComponent ()</code></td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><code>VertexComponent (SemanticName semName, IndexT semIndex, Format format)</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>GetSemanticName () const</code></td>
<td>get semantic name</td>
</tr>
<tr>
<td><code>GetSemanticIndex () const</code></td>
<td>get semantic index</td>
</tr>
<tr>
<td><code>GetFormat () const</code></td>
<td>get vertex component format</td>
</tr>
<tr>
<td><code>GetByteSize () const</code></td>
<td>get the byte size of the vertex component</td>
</tr>
<tr>
<td><code>GetSignature () const</code></td>
<td>get a unique signature of the vertex component</td>
</tr>
<tr>
<td><code>GetAccessType () const</code></td>
<td>get access type</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Util::String SemanticNameToString (SemanticName n)</code></td>
<td>convert semantic name to string</td>
</tr>
<tr>
<td><code>Util::String FormatToString (Format f)</code></td>
<td>convert format to string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Tue Feb 19 12:16:45 2008
CoreGraphics::VertexLayout
#include <vertexlayout.h>

Inheritance diagram for 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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void Setup (const Util::Array&lt;CoreGraphics::VertexComponent&gt; &amp;c)</code></td>
<td>setup the vertex layout</td>
</tr>
<tr>
<td>IDirect3DVertexDeclaration9 * <code>GetD3D9VertexDeclaration () const</code></td>
<td>get pointer to d3d9 vertex declaration</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) const</code></td>
<td>get vertex component at index</td>
</tr>
<tr>
<td><code>bool HasComponent (CoreGraphics::VertexComponent::SemanticName semName, IndexT semIndex) const</code></td>
<td>return true if vertex component exists</td>
</tr>
<tr>
<td><code>IndexT FindComponent (CoreGraphics::VertexComponent::SemanticName semName, IndexT semIndex) const</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>IndexT GetComponentByteOffset (CoreGraphics::VertexComponent::SemanticName semName, IndexT semIndex) const</code></td>
<td>get component offset from start of vertex</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>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 name</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 code</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 name</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 code</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>
Static Protected Member Functions

<table>
<thead>
<tr>
<th>static Util::String</th>
<th>BuildSignature (const Util::Array<a href="">CoreGraphics::VertexComponent</a> &amp;c)</th>
</tr>
</thead>
</table>

get sharing signature for a set of vertex components
Member Function Documentation

IndexT
Base::VertexLayoutBase::GetComponentByteOffset (CoreGraphics::VertexComponent::SemanticName
IndexT
)

get component offset from start of vertex

Returns the component offset in bytes from the beginning of a vertex
to the start of the given vertex components. Returns InvalidIndex if the
vertex component doesn't exist!

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.

dump 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:
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><strong>VertexLayoutServer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~VertexLayoutServer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>void 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 () const</strong></td>
<td>Return true if open</td>
</tr>
<tr>
<td><strong>Ptr&lt; VertexLayout &gt; CreateSharedVertexLayout (const Util::Array&lt; VertexComponent &gt; &amp;vertexComponents)</strong></td>
<td>Create shared vertex layout object</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,</td>
</tr>
<tr>
<td>by fourcc</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></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><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.
Debug::CorePageHandler
#include <corepagehandler.h>

Inheritance diagram for Debug::CorePageHandler:
Detailed Description

Provide information about Core subsystem to debug http server.

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

**CorePageHandler ()**

*constructor*

**virtual bool AcceptsRequest (const Ptr< Http::HttpRequest >&request)**

*return true if the http request is accepted by the request handler*

**virtual void HandleRequest (const Ptr< Http::HttpRequest >&request)**

*handle a http request, the handler is expected to fill the content stream with response data*

**const Util::String & GetName () const**

*get a human readable name of the request handler*

**const Util::String & GetDesc () const**

*get a human readable description of the request handler*

**const Util::String & GetRootLocation () const**

*get a resource location path which is accepted by the handler (e.g. "index.html")*

**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,*
<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>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>SetName</code></td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><code>SetDesc</code></td>
<td>set human readable description</td>
</tr>
<tr>
<td><code>SetRootLocation</code></td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
Member Function Documentation

```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]
```

class name

Get the class name of the object.

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

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>virtual bool AcceptsRequest (const Ptr<a href="Http::HttpRequest">Http::HttpRequest</a> &amp;request)</strong></td>
<td>return true if the http request is accepted by the request handler</td>
</tr>
<tr>
<td><strong>virtual void HandleRequest (constPtr<a href="Http::HttpRequest">Http::HttpRequest</a> &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;/index.html&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,</td>
</tr>
<tr>
<td>by string</td>
<td></td>
</tr>
<tr>
<td>-----------</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 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><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>void</th>
<th><strong>SetName</strong> (const Util::String &amp;n)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set human readable name of the request handler</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetDesc</strong> (const Util::String &amp;d)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set human readable description</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>SetRootLocation</strong> (const Util::String &amp;l)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set the root location of the request handler</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.
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><code>IoPageHandler()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>AcceptsRequest(r)</code></td>
<td>virtual bool; return true if the http request is accepted by the request handler</td>
</tr>
<tr>
<td><code>HandleRequest(r)</code></td>
<td>virtual void; handle a http request, the handler is expected to fill the content stream with response data</td>
</tr>
<tr>
<td><code>GetName()</code></td>
<td>const Util::String &amp;; get a human readable name of the request handler</td>
</tr>
<tr>
<td><code>GetDesc()</code></td>
<td>const Util::String &amp;; get a human readable description of the request handler</td>
</tr>
<tr>
<td><code>GetRootLocation()</code></td>
<td>const Util::String &amp;; get a resource location path which is accepted by the handler (e.g. &quot;/index.html&quot;)</td>
</tr>
<tr>
<td><code>GetRefCount()</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(r)</code></td>
<td>bool; return true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf(className)</code></td>
<td>bool; return true if this object is instance of given class by string</td>
</tr>
<tr>
<td><code>IsInstanceOf(classFourCC)</code></td>
<td>bool; return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td><code>IsA()</code></td>
<td>bool; return true if this object is instance of given class, or a derived class</td>
</tr>
</tbody>
</table>

```cpp
virtual bool AcceptsRequest(const Http::HttpRequest &request)
return true if the http request is accepted by the request handler
```

```cpp
virtual void HandleRequest(const Http::HttpRequest &request)
handle a http request, the handler is expected to fill the content stream with response data
```

```cpp
const Util::String & GetName() const
get a human readable name of the request handler
```

```cpp
const Util::String & GetDesc() const
get a human readable description of the request handler
```

```cpp
const Util::String & GetRootLocation() const
get a resource location path which is accepted by the handler (e.g. "/index.html")
```

```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

| 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>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
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::MemoryPageHandler
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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemoryPageHandler ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>AcceptsRequest (const Ptr&lt; Http::HttpRequest &amp;request)</strong></td>
<td>return true if the http request is accepted by the request handler</td>
</tr>
<tr>
<td><strong>HandleRequest (const Ptr&lt; 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;index.html&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,</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) 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>
<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>void <strong>SetName</strong> (const Util::String &amp;n)</td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td>void <strong>SetDesc</strong> (const Util::String &amp;d)</td>
<td>set human readable description</td>
</tr>
<tr>
<td>void <strong>SetRootLocation</strong> (const Util::String &amp;l)</td>
<td>set the root location of the request handler</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::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>virtual bool <strong>AcceptsRequest</strong> (const Ptr<a href="Http::HttpRequest">Http::HttpRequest</a> &amp;request)</td>
<td>return true if the http request is accepted by the request handler</td>
</tr>
<tr>
<td>virtual void <strong>HandleRequest</strong> (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; <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;index.html&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,</td>
</tr>
<tr>
<td>by string</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const Util::FourCC &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 Util::String &amp; <strong>GetClassName</strong> () const</td>
<td></td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td>Util::FourCC <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><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 <strong>SetName</strong> (const Util::String &amp;n)</td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td>void <strong>SetDesc</strong> (const Util::String &amp;d)</td>
<td>set human readable description</td>
</tr>
<tr>
<td>void <strong>SetRootLocation</strong> (const Util::String &amp;l)</td>
<td>set the root location of the request handler</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.
Debug::MiniDump
#include <minidump.h>

Inheritance diagram for 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
Static Public Member Functions

static bool WriteMiniDump ()

write a mini dump
Member Function Documentation

bool Win32::Win32MiniDump::WriteMiniDump() [static, inherited]

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

Debug::ScriptingPageHandler
#include <scriptingpagehandler.h>

Inheritance diagram for Debug::ScriptingPageHandler:
Detailed Description

Provide information about **Scripting** 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>ScriptingPageHandler</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual bool <strong>AcceptsRequest</strong> (const Ptr<a href="Http::HttpRequest">Http::HttpRequest</a> &amp;request)</td>
<td>return true if the http request is accepted by the request handler</td>
</tr>
<tr>
<td>virtual void <strong>HandleRequest</strong> (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; <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;index.html&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,</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>bool (const <code>Util::FourCC</code> &amp;rttiFourCC) 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 <code>Util::String</code> &amp; get the class name</td>
</tr>
<tr>
<td><code>GetClassFourCC</code></td>
<td><code>Util::FourCC</code> ( ) 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>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>SetName</strong></td>
<td>(const Util::String &amp;n) <strong>set human readable name of the request handler</strong></td>
</tr>
<tr>
<td><strong>SetDesc</strong></td>
<td>(const Util::String &amp;d) <strong>set human readable description</strong></td>
</tr>
<tr>
<td><strong>SetRootLocation</strong></td>
<td>(const Util::String &amp;l) <strong>set the root location of the request handler</strong></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::ShaderPageHandler
Debug::ShaderPageHandler Class Reference

#include <shaderpagehandler.h>

Inheritance diagram for 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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShaderPageHandler ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual bool AcceptsRequest (const Ptr<a href="Http::HttpRequest">Http::HttpRequest</a> &amp;request)</strong></td>
<td>return true if the http request is accepted by the request handler</td>
</tr>
<tr>
<td><strong>virtual void HandleRequest (const Ptr<a href="Http::HttpRequest">Http::HttpRequest</a> &amp;request)</strong></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><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.</td>
</tr>
<tr>
<td></td>
<td>&quot;index.html&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,</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------</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>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>void <strong>SetName</strong> (const Util::String &amp;n)</td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td>void <strong>SetDesc</strong> (const Util::String &amp;d)</td>
<td>set human readable description</td>
</tr>
<tr>
<td>void <strong>SetRootLocation</strong> (const Util::String &amp;l)</td>
<td>set the root location of the request handler</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::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.


(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 bool <strong>AcceptsRequest</strong> (const Ptr<a href="Http::HttpRequest">Http::HttpRequest</a> &amp;request)</td>
<td>return true if the http request is accepted by the request handler</td>
</tr>
<tr>
<td>virtual void <strong>HandleRequest</strong> (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; <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;index.html&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,</td>
</tr>
<tr>
<td>Method</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>
### 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>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <strong>SetName</strong></td>
<td>(const Util::String &amp;n)</td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td>void <strong>SetDesc</strong></td>
<td>(const Util::String &amp;d)</td>
<td>set human readable description</td>
</tr>
<tr>
<td>void <strong>SetRootLocation</strong></td>
<td>(const Util::String &amp;l)</td>
<td>set the root location of the request handler</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.
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</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 adapter)</strong></td>
<td>Check if the adapter actually 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>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>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HWND <code>GetHwnd () const</code></td>
<td>get the application window HWND</td>
</tr>
<tr>
<td>CoreGraphics::Adapter::Code void <code>SetAdapter (CoreGraphics::Adapter::Code a)</code></td>
<td>set display adapter (make sure adapter exists!)</td>
</tr>
<tr>
<td>CoreGraphics::Adapter::Code <code>GetAdapter () const</code></td>
<td>get display adapter</td>
</tr>
<tr>
<td>CoreGraphics::Adapter::Code void <code>SetDisplayMode (const CoreGraphics::DisplayMode &amp;m)</code></td>
<td>set display mode (make sure the display mode is supported!)</td>
</tr>
<tr>
<td>CoreGraphics::DisplayMode &amp; <code>GetDisplayMode () const</code></td>
<td>get display mode</td>
</tr>
<tr>
<td>CoreGraphics::AntiAliasQuality::Code void <code>SetAntiAliasQuality (CoreGraphics::AntiAliasQuality::Code aa)</code></td>
<td>set antialias quality</td>
</tr>
<tr>
<td>CoreGraphics::AntiAliasQuality::Code <code>GetAntiAliasQuality () const</code></td>
<td>get antialias quality</td>
</tr>
<tr>
<td>CoreGraphics::AntiAliasQuality::Code void <code>SetFullScreen (bool b)</code></td>
<td>set windowed/fullscreen mode</td>
</tr>
<tr>
<td>bool <code>IsFullScreen () const</code></td>
<td>get windowed/fullscreen mode</td>
</tr>
<tr>
<td>CoreGraphics::DisplayModeSwitchEnabled::Code void <code>SetDisplayModeSwitchEnabled (bool b)</code></td>
<td>enable display mode switch when running fullscreen (default is true);</td>
</tr>
<tr>
<td>bool <code>IsDisplayModeSwitchEnabled () const</code></td>
<td>is display mode switch enabled for fullscreen?</td>
</tr>
<tr>
<td>CoreGraphics::TripleBufferingEnabled::Code void <code>SetTripleBufferingEnabled (bool b)</code></td>
<td>enable triple buffer for fullscreen (default is double buffering)</td>
</tr>
<tr>
<td>bool <code>IsTripleBufferingEnabled () const</code></td>
<td>is triple buffer enabled for fullscreen?</td>
</tr>
<tr>
<td>CoreGraphics::AlwaysOnTop::Code void <code>SetAlwaysOnTop (bool b)</code></td>
<td>set always-on-top behaviour</td>
</tr>
<tr>
<td>bool <code>IsAlwaysOnTop () const</code></td>
<td>get always-on-top behaviour</td>
</tr>
</tbody>
</table>
void SetVerticalSyncEnabled (bool b)  
\hspace*{1em} turn vertical sync on/off

bool IsVerticalSyncEnabled () const  
\hspace*{1em} get vertical sync flag

void SetIconName (const Util::String &s)  
\hspace*{1em} set optional window icon resource name

const Util::String & GetIconName () const  
\hspace*{1em} get optional window icon resource name

void SetWindowTitle (const Util::String &t)  
\hspace*{1em} set window title string (can be changed anytime)

const Util::String & GetWindowTitle () const  
\hspace*{1em} get window title string

bool isOpen () const  
\hspace*{1em} return true if display is currently open

void AttachEventHandler (const Ptr<CoreGraphics::DisplayEventHandler> &h)  
\hspace*{1em} attach a display event handler

void RemoveEventHandler (const Ptr<CoreGraphics::DisplayEventHandler> &h)  
\hspace*{1em} remove a display event handler

int GetRefCount () const  
\hspace*{1em} get the current refcount

void AddRef ()  
\hspace*{1em} increment refcount by one

void Release ()  
\hspace*{1em} decrement refcount and destroy object if refcount is zero

bool IsInstanceOf (const Rtti &rtti) const  
\hspace*{1em} return true if this object is instance of given class

bool IsInstanceOf (const Util::String &className) const  
\hspace*{1em} return true if this object is instance of given class by string

bool IsInstanceOf (const Util::FourCC &classFourCC) const  
\hspace*{1em} return true if this object is instance of given class by
<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsA</td>
<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></td>
<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></td>
<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>GetClassName</td>
<td>const Util::String &amp; GetClassName () const</td>
<td>get the class name.</td>
</tr>
<tr>
<td>GetClassFourCC</td>
<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>virtual bool OpenWindow</td>
<td>open the application window</td>
</tr>
<tr>
<td>virtual void CloseWindow</td>
<td>close the application window</td>
</tr>
<tr>
<td>virtual void OnMinimized</td>
<td>called on WM_SIZE when window is minimized</td>
</tr>
<tr>
<td>virtual void OnRestored</td>
<td>called on WM_SIZE when window is restored</td>
</tr>
<tr>
<td>virtual bool OnSetCursor</td>
<td>called on WM_SETCURSOR</td>
</tr>
<tr>
<td>virtual void OnPaint</td>
<td>called on WM_PAINT</td>
</tr>
<tr>
<td>virtual void OnSetFocus</td>
<td>called on WM_SETFOCUS</td>
</tr>
<tr>
<td>virtual void OnKillFocus</td>
<td>called on WM_KILLFOCUS</td>
</tr>
<tr>
<td>virtual void OnCloseRequested</td>
<td>called on WM_CLOSE to request if window should be closed</td>
</tr>
<tr>
<td>virtual void OnToggleFullscreenWindowed</td>
<td>called when Alt-Enter is pressed</td>
</tr>
<tr>
<td>virtual void OnKeyDown</td>
<td>called on WM_KEYDOWN</td>
</tr>
<tr>
<td>virtual void OnKeyUp</td>
<td>called on WM_KEYUP</td>
</tr>
<tr>
<td>virtual void OnChar</td>
<td>called on WM_CHAR</td>
</tr>
<tr>
<td>virtual void OnMouseButton</td>
<td>called on mouse button event</td>
</tr>
<tr>
<td>virtual void OnMouseMove</td>
<td>called on WM_MOUSEMOVE</td>
</tr>
<tr>
<td>virtual void OnMouseWheel</td>
<td>called on WM_MOUSEWHEEL</td>
</tr>
<tr>
<td>Input::Key::Code TranslateKeyCode</td>
<td>(WPARAM wParam) const</td>
</tr>
<tr>
<td>Class</td>
<td>Function Name</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Math::float2</td>
<td><strong>ComputeAbsMousePos</strong> (LPARAM lParam) const</td>
</tr>
<tr>
<td>Math::float2</td>
<td><strong>ComputeNormMousePos</strong> (const Math::float2 &amp;absMousePos) const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>NotifyEventHandlers</strong> (const CoreGraphics::DisplayEvent &amp;e)</td>
</tr>
</tbody>
</table>
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

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.

```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(HWND hWnd, 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 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.

```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::D3D9IndexBuffer
Direct3D9::D3D9IndexBuffer Class Reference

#include <d3d9indexbuffer.h>

Inheritance diagram for Direct3D9::D3D9IndexBuffer:
Detailed Description

D3D9 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>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>Method</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><strong>IDirect3DIndexBuffer9 * GetD3D9IndexBuffer () const</strong></td>
<td>Get d3d9 index buffer pointer</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>SizeT GetNumIndices () const</strong></td>
<td>Get number of indices</td>
</tr>
<tr>
<td><strong>Usage GetUsage () const</strong></td>
<td>Get resource usage 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>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 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>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</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()</code></td>
<td>get optional resource saver</td>
</tr>
<tr>
<td><code>GetUseCount()</code></td>
<td>get current use count</td>
</tr>
<tr>
<td><code>Load()</code></td>
<td>load the resource</td>
</tr>
<tr>
<td><code>GetState()</code></td>
<td>get current state</td>
</tr>
<tr>
<td><code>IsLoaded()</code></td>
<td>return true if current state is Loaded</td>
</tr>
<tr>
<td><code>IsPending()</code></td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td><code>LoadFailed()</code></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></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>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

| 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>void SetIndexType (CoreGraphics::IndexType::Code type)</td>
<td>set the index type (Index16 or Index32)</td>
</tr>
<tr>
<td>void SetNumIndices (SizeT num)</td>
<td>set number of indices</td>
</tr>
<tr>
<td>void SetUsage (Usage usage)</td>
<td>set resource usage type</td>
</tr>
<tr>
<td>void SetAccess (Access access)</td>
<td>set resource cpu access type</td>
</tr>
<tr>
<td>void SetState (State s)</td>
<td>set current state</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>
Member Function Documentation

`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.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Direct3D9::D3D9MemoryIndexBufferLoader
Direct3D9::D3D9MemoryIndexBufferLoader

Class Reference

#include <d3d9memoryindexbufferloader.h>

Inheritance diagram for Direct3D9::D3D9MemoryIndexBufferLoader:
Detailed Description

Initialize a **D3D9IndexBuffer** from data in memory. 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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool OnLoadRequested ()</td>
<td>Called by resource when a load is requested</td>
</tr>
<tr>
<td>void Setup (CoreGraphics::IndexType::Code indexType, SizeT numIndices, void *ptr, SizeT numBytes)</td>
<td>Setup index buffer data, must remain valid until OnLoadRequested() is called!</td>
</tr>
<tr>
<td>void Setup (CoreGraphics::IndexType::Code indexType, SizeT numIndices, SizeT numBytes, CoreGraphics::IndexBuffer::Usage usage, CoreGraphics::IndexBuffer::Access access)</td>
<td>Setup a empty index buffer</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>virtual bool CanLoadAsync () const</td>
<td>Return true if asynchronous loading is supported</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>
</tbody>
</table>

Resource::State
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetState () const</code></td>
<td>return current 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><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 Direct3D9::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). The data pointer provided to Setup() will be invalidated inside OnLoadRequested(). Usage will be set to UsageImmutable and Access to AccessNone.

Reimplemented from Resources::ResourceLoader.

void Base::MemoryIndexBufferLoaderBase::Setup(
    CoreGraphics::IndexType::Code indexType,
    SizeT num,
    void * ptr,
    SizeT numBytes
) [inherited]

setup index buffer data, must remain valid until OnLoadRequested() is called!

Setup all information needed to initialize the IndexBuffer resource. The data must remain valid until OnLoadRequested() is called (which will invalidate the data).

void Base::MemoryIndexBufferLoaderBase::Setup(
    CoreGraphics::IndexType::Code indexType,
    SizeT num,
    SizeT numBytes,
    CoreGraphics::IndexBuffer::Usage usage,
    CoreGraphics::IndexBuffer::Access access
)

setup a empty index buffer

Setup all information needed to initialize a empty IndexBuffer
void Base::MemoryIndexBufferLoaderBase::Setup(CoreGraphics::IndexType::Code type, SizeT num, void *ptr, SizeT numBytes, CoreGraphics::IndexBuffer::Usage usage, CoreGraphics::IndexBuffer::Access access)

setup index buffer data, must remain valid until `OnLoadRequested()` is called!

Setup all information needed to initialize a `IndexBuffer` resource.

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`, `Direct3D9::D3D9StreamTextureLoader`, `CoreGraphics::StreamAnimationLoader`, `CoreGraphics::StreamMeshLoader`, and `Models::StreamModelLoader`.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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::D3D9MemoryVertexBufferLoader
Direct3D9::D3D9MemoryVertexBufferLoader Class Reference

#include <d3d9memoryvertexbufferloader.h>

Inheritance diagram for Direct3D9::D3D9MemoryVertexBufferLoader:
Detailed Description

Initialize a D3D9VertexBuffer from data in memory. 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 void OnLoadRequested ()</td>
<td>Called by resource when a load is requested</td>
</tr>
<tr>
<td>void Setup (const Util::Array<a href="">CoreGraphics::VertexComponent</a> &amp;vertexComponents, SizeT numVertices, void *ptr, SizeT numBytes)</td>
<td>Setup vertex buffer data, must remain valid until OnLoadRequested() is called!</td>
</tr>
<tr>
<td>void Setup (const Util::Array<a href="">CoreGraphics::VertexComponent</a> &amp;vertexComponents, SizeT numVertices, SizeT numBytes, CoreGraphics::VertexBuffer::Usage usage, CoreGraphics::VertexBuffer::Access access)</td>
<td>Setup a vertex buffer</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>virtual bool CanLoadAsync () const</td>
<td>Return true if asynchronous loading is supported</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</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 <code>Resource</code> into loaded state</td>
</tr>
<tr>
<td><code>Resource::State</code></td>
<td><code>GetState()</code> const return current state</td>
</tr>
<tr>
<td><code>int</code> <code>GetRefCount</code></td>
<td>const get the current refcount</td>
</tr>
<tr>
<td><code>void</code> <code>AddRef</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td><code>void</code> <code>Release</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td><code>bool</code> <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>bool</code> <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><code>bool</code> <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>bool</code> <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>bool</code> <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>bool</code> <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>const </code> <code>Util::String</code> &amp;</td>
<td><code>GetClassName()</code> const get the class name</td>
</tr>
<tr>
<td><code>Util::FourCC</code> &amp;</td>
<td><code>GetClassFourCC()</code> const 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

```c
void SetState (Resource::State S)
set current state
```
Member Function Documentation

```cpp
bool Direct3D9::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). The data pointer provided to Setup() will be invalidated inside OnLoadRequested(). Resource usage will be set to UsageImmutable and resource access to AccessNone.

Reimplemented from Resources::ResourceLoader.

```cpp
void Base::MemoryVertexBufferLoaderBase::Setup(
    const Util::Array<CoreGraphics::VertexComponent>& components, 
    SizeT num, 
    void* ptr, 
    SizeT numBytes
)

```cpp
void Base::MemoryVertexBufferLoaderBase::Setup(
    const Util::Array<CoreGraphics::VertexComponent>& vertexComponents, 
    SizeT num, 
    SizeT numBytes, 
    CoreGraphics::VertexBuffer::Usage usage, 
    CoreGraphics::VertexBuffer::Access access
)
```

setup vertex buffer data, must remain valid until OnLoadRequested() is called!

Setup all information needed to initialize the VertexBuffer resource. The data must remain valid until OnLoadRequested() is called (which will invalidate the data).
setup a empty vertex buffer

Setup all information needed to initialize a empty VertexBuffer resource.

```cpp
void Base::MemoryVertexBufferLoaderBase::Setup(
    const Util::Array<
        CoreGraphics::VertexComponent>
        & components,
    SizeT num, 
    void * ptr,
    SizeT numBytes,
    CoreGraphics::VertexBuffer::Usage usage,
    CoreGraphics::VertexBuffer::Access access
)
```

setup a vertex buffer, vertex buffer data, must remain valid until
**OnLoadRequested()** is called!

Setup all information needed to initialize the VertexBuffer resource. The data must remain valid until **OnLoadRequested()** is called (which will invalidate the data).

```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**, **Direct3D9::D3D9StreamTextureLoader**, **CoreGraphics::StreamAnimationLoader**, **CoreGraphics::StreamMeshLoader**, and **Models::StreamModelLoader**.
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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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::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><code>D3D9RenderDevice()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~D3D9RenderDevice()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>IDirect3DDevice9 * GetDirect3DDevice()</code></td>
<td>Get pointer to the d3d device</td>
</tr>
<tr>
<td><code>bool Open()</code></td>
<td>Open the device</td>
</tr>
<tr>
<td><code>void Close()</code></td>
<td>Close the device</td>
</tr>
<tr>
<td><code>bool BeginFrame()</code></td>
<td>Begin complete frame</td>
</tr>
<tr>
<td><code>void SetVertexBuffer(const Ptr&lt;CoreGraphics::VertexBuffer&gt; &amp;vb)</code></td>
<td>Set current vertex buffer</td>
</tr>
<tr>
<td><code>void SetIndexBuffer(const Ptr&lt;CoreGraphics::IndexBuffer&gt; &amp;ib)</code></td>
<td>Set current index buffer</td>
</tr>
<tr>
<td><code>void Draw()</code></td>
<td>Draw current primitives</td>
</tr>
<tr>
<td><code>void EndPass()</code></td>
<td>End current pass</td>
</tr>
<tr>
<td><code>void EndFrame()</code></td>
<td>End complete frame</td>
</tr>
<tr>
<td><code>void Present()</code></td>
<td>Present the rendered scene</td>
</tr>
<tr>
<td><code>void 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 IsOpen()</code></td>
<td>Return true if currently open</td>
</tr>
<tr>
<td><code>AttachEventHandler(const Ptr&lt;</code></td>
<td>Attach event handler</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void CoreGraphics::RenderEventHandler &amp;h)</code></td>
<td>attach a render event handler</td>
</tr>
<tr>
<td>`void RemoveEventHandler (const Ptr&lt;</td>
<td>remove a render event handler</td>
</tr>
<tr>
<td>CoreGraphics::RenderEventHandler &amp;h)`</td>
<td></td>
</tr>
<tr>
<td><code>const Ptr &lt; CoreGraphics::RenderTarget &gt; &amp;</code></td>
<td><code>GetDefaultRenderTarget () const</code></td>
</tr>
<tr>
<td></td>
<td>get default render target</td>
</tr>
<tr>
<td>`void BeginPass (const Ptr&lt; CoreGraphics::</td>
<td>begin rendering a frame pass</td>
</tr>
<tr>
<td>RenderTarget &gt; &amp;rt, const Ptr&lt;</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::ShaderInstance &gt; &amp;passShader)</td>
<td></td>
</tr>
<tr>
<td>`void BeginBatch (CoreGraphics::BatchType::</td>
<td>begin rendering a batch inside</td>
</tr>
<tr>
<td>Code batchType, const Ptr&lt;</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::ShaderInstance &gt; &amp;batchShader)</td>
<td></td>
</tr>
<tr>
<td><code>const Ptr &lt; CoreGraphics::VertexBuffer &gt; &amp;</code></td>
<td><code>GetVertexBuffer () const</code></td>
</tr>
<tr>
<td></td>
<td>get current vertex buffer</td>
</tr>
<tr>
<td><code>const Ptr &lt; CoreGraphics::IndexBuffer &gt; &amp;</code></td>
<td><code>GetIndexBuffer () const</code></td>
</tr>
<tr>
<td></td>
<td>get current index buffer</td>
</tr>
<tr>
<td>`void SetPrimitiveGroup (const CoreGraphics::</td>
<td>set current primitive group</td>
</tr>
<tr>
<td>PrimitiveGroup &amp;pg)`</td>
<td></td>
</tr>
<tr>
<td><code>const CoreGraphics::PrimitiveGroup &amp;</code></td>
<td><code>GetPrimitiveGroup () const</code></td>
</tr>
<tr>
<td></td>
<td>get current primitive group</td>
</tr>
<tr>
<td><code>void EndBatch ()</code></td>
<td>end current batch</td>
</tr>
<tr>
<td><code>bool IsInBeginFrame () const</code></td>
<td>check if inside BeginFrame</td>
</tr>
<tr>
<td><code>int GetRefCount () const</code></td>
<td></td>
</tr>
</tbody>
</table>
Get the current refcount

```c++
void AddRef ()
    increment refcount by one

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

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
```

```c++
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
```

```c++
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>
Protected Member Functions

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

*notify event handlers about an event*
Member Function Documentation

```cpp
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
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
get pointer to the d3d device

Return a pointer to d3d device. Asserts that the device exists.
```

```cpp
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 ( )
```
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::SetVertexBuffer(const Ptr<CoreGraphics::VertexBuffer> & vb)
```

**set current vertex buffer**

Sets the vertex buffer to use 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::EndPass()
```

draw current pass

End the current rendering pass. This will flush all texture stages in order to keep the d3d9 resource reference consistent without too much hassle.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
void Direct3D9::D3D9RenderDevice::EndFrame()
```

draw complete frame

End a complete frame. Call this once per frame after rendering and presentation has happened, and only if `BeginFrame()` returns true.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
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**.

```cpp
void Direct3D9::D3D9RenderDevice::SaveScreenshot( CoreGraphics::ImageFileFormat::Code fmt,
```
save a screenshot to the provided stream

Save the backbuffer to the provided stream.

Reimplemented from **Base::RenderDeviceBase**.

```cpp
void Base::RenderDeviceBase::AttachEventHandler(const Ptr<CoreGraphics::RenderEventHandler> & h) [inherited]
```

attach a render event handler

Attach an event handler to the render device.

```cpp
void Base::RenderDeviceBase::RemoveEventHandler(const Ptr<CoreGraphics::RenderEventHandler> & h) [i]
```

remove a render event handler

Remove an event handler from the display device.

```cpp
bool Base::RenderDeviceBase::NotifyEventHandlers(const CoreGraphics::RenderEvent & 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.
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 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>D3D9RenderTarget()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~D3D9RenderTarget()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Setup()</code></td>
<td>Setup the render target object</td>
</tr>
<tr>
<td><code>Discard()</code></td>
<td>Discard the render target object</td>
</tr>
<tr>
<td><code>BeginPass()</code></td>
<td>Begin a render pass</td>
</tr>
<tr>
<td><code>EndPass()</code></td>
<td>End current render pass</td>
</tr>
<tr>
<td><code>GenerateMipLevels()</code></td>
<td>Generate mipmap levels</td>
</tr>
<tr>
<td><code>IsDefaultRenderTarget()</code> const</td>
<td>Get default render target flag</td>
</tr>
<tr>
<td><code>HasColorBuffer(IndexT colorBufferIndex) const</code></td>
<td>Return true if color buffer exists</td>
</tr>
<tr>
<td><code>HasDepthStencilBuffer()</code> const</td>
<td>Return true if the render target has a depth/stencil buffer</td>
</tr>
<tr>
<td><code>IsValid()</code></td>
<td>Return true if valid (has been setup)</td>
</tr>
<tr>
<td><code>setWidth(SizeT w)</code></td>
<td>Set render target width</td>
</tr>
<tr>
<td><code>GetWidth()</code> const</td>
<td>Get width of render target in pixels</td>
</tr>
<tr>
<td><code>SetHeight(SizeT h)</code></td>
<td>Set render target height</td>
</tr>
<tr>
<td><code>GetHeight()</code> const</td>
<td>Get height of render target in pixels</td>
</tr>
<tr>
<td><code>SetAntiAliasQuality</code></td>
<td></td>
</tr>
<tr>
<td>Function Call</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void (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 AddColorBuffer (CoreGraphics::PixelFormat::Code colorFormat)</code></td>
<td>add a color buffer</td>
</tr>
<tr>
<td><code>GetSizeT GetNumColorBuffers () const</code></td>
<td>get number of color buffers</td>
</tr>
<tr>
<td><code>CoreGraphics::PixelFormat::Code GetColorBufferFormat (IndexT colorBufferIndex) const</code></td>
<td>get color buffer format at index</td>
</tr>
<tr>
<td><code>void AddDepthStencilBuffer ()</code></td>
<td>add 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 SetResolveTextureWidth (SizeT w)</code></td>
<td>set resolve texture width</td>
</tr>
<tr>
<td><code>GetSizeT 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>GetSizeT GetResolveTextureHeight () const</code></td>
<td>get resolve texture height</td>
</tr>
<tr>
<td><code>void SetClearColor (const Math::float4 &amp;c)</code></td>
<td>set clear color</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetClearColor () const</code></td>
<td></td>
</tr>
</tbody>
</table>
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<uint> &r)
set the current resolve rectangle (in pixels)

const Math::rectangle<uint> & 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

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, 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>
</table>

*dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)*
<table>
<thead>
<tr>
<th>static const IndexT</th>
<th><strong>MaxNumColorBuffers</strong> = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>max number of color buffers</td>
<td></td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><code>SetupMultiSampleType</code> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>setup compatible multisample type</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><code>SetDefaultRenderTarget</code> (bool b)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set to true if default render target</em></td>
</tr>
</tbody>
</table>
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.
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><strong>State</strong></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>D3D9Shader ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <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 () const</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 (const Ptr<a href="">CoreGraphics::ShaderInstance</a> &amp;inst)</strong></td>
<td>discard a shader instance</td>
</tr>
<tr>
<td><strong>GetAllShaderInstances () const</strong></td>
<td>get all instances</td>
</tr>
<tr>
<td>void <strong>SetAsyncEnabled (bool b)</strong></td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td>bool <strong>IsAsyncEnabled () const</strong></td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td>void <strong>SetResourceId (const ResourceId &amp;id)</strong></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td>const ResourceId &amp; <strong>GetResourceId () const</strong></td>
<td>get the resource identifier</td>
</tr>
<tr>
<td>void <strong>SetLoader (const Ptr&lt;ResourceLoader&gt; &amp;loader)</strong></td>
<td>set optional resource loader</td>
</tr>
<tr>
<td>const Ptr&lt;ResourceLoader&gt; &amp; <strong>GetLoader () const</strong></td>
<td>get optional resource loader</td>
</tr>
</tbody>
</table>
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

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
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetState(State S)</code></td>
<td>set current state</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

**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.
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><strong>D3D9ShaderInstance ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~D3D9ShaderInstance ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>ID3DXEffect * GetD3D9Effect ( ) const</strong></td>
<td>get pointer to d3d9 effect object</td>
</tr>
<tr>
<td><strong>bool SelectActiveVariation (CoreGraphics::ShaderFeature::Mask featureMask)</strong></td>
<td>select active variation by feature mask</td>
</tr>
<tr>
<td><strong>SizeT Begin ( )</strong></td>
<td>begin rendering through the currently selected variation, returns no. passes</td>
</tr>
<tr>
<td><strong>void BeginPass (IndexT passIndex)</strong></td>
<td>begin pass</td>
</tr>
<tr>
<td><strong>void Commit ( )</strong></td>
<td>commit changes before rendering</td>
</tr>
<tr>
<td><strong>void EndPass ( )</strong></td>
<td>end pass</td>
</tr>
<tr>
<td><strong>void End ( )</strong></td>
<td>end rendering through variation</td>
</tr>
<tr>
<td><strong>void Discard ( )</strong></td>
<td>discard the shader instance, must be called when instance no longer needed</td>
</tr>
<tr>
<td><strong>bool IsValid ( ) const</strong></td>
<td>return true if this object is valid</td>
</tr>
<tr>
<td><strong>const Ptr &lt; CoreGraphics::Shader &gt; &amp; GetOriginalShader ( ) const</strong></td>
<td>get pointer to original shader which created this instance</td>
</tr>
<tr>
<td><strong>bool HasVariableByName (const CoreGraphics::ShaderVariable::Name &amp;n) const</strong></td>
<td>return true if the shader instance has a variable by name</td>
</tr>
<tr>
<td><strong>bool HasVariableBySemantic (const</strong></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>return true if shader has variable by semantic</code></td>
<td></td>
</tr>
<tr>
<td><code>get number of variables</code></td>
<td></td>
</tr>
<tr>
<td><code>get a variable by index</code></td>
<td></td>
</tr>
<tr>
<td><code>get a variable by name</code></td>
<td></td>
</tr>
<tr>
<td><code>get a variable by semantic</code></td>
<td></td>
</tr>
<tr>
<td><code>return true if variation exists by matching feature mask</code></td>
<td></td>
</tr>
<tr>
<td><code>get number of variations in the shader</code></td>
<td></td>
</tr>
<tr>
<td><code>get shader variation by index</code></td>
<td></td>
</tr>
<tr>
<td><code>get shader variation by feature mask</code></td>
<td></td>
</tr>
<tr>
<td><code>get currently active variation</code></td>
<td></td>
</tr>
<tr>
<td><code>add a pre-shader</code></td>
<td></td>
</tr>
<tr>
<td><code>remove a pre-shader</code></td>
<td></td>
</tr>
</tbody>
</table>

```cpp
const Util::Array
```
< Ptr < CoreGraphics::PreShader > > &

- **GetPreShaders** () const
  - get array of pre-shaders

- **GetRefCount** () const
  - get the current refcount

- **AddRef** ()
  - increment refcount by one

- **Release** ()
  - decrement refcount and destroy object if refcount is zero

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

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

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

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

- **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 &rttiFourCC) const
  - return true if this object is instance of given class, or a derived class, by fourcc

- **GetClassName** () const
  - get the class name

- **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>virtual void Setup(const Ptr&lt; CoreGraphics::Shader &gt; &amp;origShader)</td>
<td>setup the shader instance from its original shader object</td>
</tr>
<tr>
<td>virtual void Cleanup()</td>
<td>cleanup the shader instance</td>
</tr>
<tr>
<td>void OnLostDevice()</td>
<td>called by d3d9 shader server when d3d9 device is lost</td>
</tr>
<tr>
<td>void OnResetDevice()</td>
<td>called by d3d9 shader server when d3d9 device is reset</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
template<typename Shader>
void Direct3D9::D3D9ShaderInstance::Setup(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`.

```cpp
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.

```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]
```

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 Types

<table>
<thead>
<tr>
<th>enum</th>
<th><code>ShaderParamBindMode</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>shader parameter bind modes</em></td>
</tr>
</tbody>
</table>
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3D9ShaderServer ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~D3D9ShaderServer ()</td>
<td>destructor</td>
</tr>
<tr>
<td>bool Open ()</td>
<td>open the shader server</td>
</tr>
<tr>
<td>void Close ()</td>
<td>close the shader server</td>
</tr>
<tr>
<td>bool HasSharedVariableByName (const CoreGraphics::ShaderVariable::Name &amp;name) const</td>
<td>return true if a shared variable exists by name</td>
</tr>
<tr>
<td>bool HasSharedVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &amp;sem) const</td>
<td>return true if a shared variable exists by semantic</td>
</tr>
<tr>
<td>SizeT GetNumSharedVariables () const</td>
<td>get number of shared variables</td>
</tr>
<tr>
<td>const Ptr &lt; CoreGraphics::ShaderVariable &gt; &amp; GetSharedVariableByIdx (IndexT i)</td>
<td>get a shared variable by index</td>
</tr>
<tr>
<td>const Ptr &lt; CoreGraphics::ShaderVariable &gt; &amp; GetSharedVariableByName (const CoreGraphics::ShaderVariable::Name &amp;name) const</td>
<td>get a shared variable by name</td>
</tr>
<tr>
<td>const Ptr &lt; CoreGraphics::ShaderVariable &gt; &amp; GetSharedVariableBySemantic (const CoreGraphics::ShaderVariable::Semantic &amp;sem) const</td>
<td>get a shared variable by semantic</td>
</tr>
<tr>
<td>ID3DXEffectPool * GetD3D9EffectPool () const</td>
<td>get pointer to global effect pool</td>
</tr>
<tr>
<td>void SetShaderParamBindMode (ShaderParamBindMode m)</td>
<td>set shader param bind mode (by name or by semantic,</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td><code>default is by semantic)</code></td>
<td></td>
</tr>
<tr>
<td><code>ShaderParamBindMode</code></td>
<td><code>GetShaderParamBindMode() const</code></td>
</tr>
<tr>
<td></td>
<td><code>get shader param bind mode</code></td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsOpen() const</code></td>
</tr>
<tr>
<td></td>
<td><code>return true if the shader server is open</code></td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>HasShader(const Resources::ResourceId &amp;resId) const</code></td>
</tr>
<tr>
<td></td>
<td><code>return true if a shader exists</code></td>
</tr>
<tr>
<td><code>Ptr&lt;CoreGraphics::ShaderInstance&gt;</code></td>
<td><code>CreateShaderInstance(const Resources::ResourceId &amp;resId)</code></td>
</tr>
<tr>
<td></td>
<td><code>create a new shader instance</code></td>
</tr>
<tr>
<td><code>const Util::Dictionary&lt;Resources::ResourceId, Ptr&lt;CoreGraphics::Shader&gt; &gt; &amp;</code></td>
<td><code>GetAllShaders() const</code></td>
</tr>
<tr>
<td></td>
<td><code>get all loaded shaders</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>SetActiveShaderInstance(const Ptr&lt;CoreGraphics::ShaderInstance&gt; &amp;shaderInst)</code></td>
</tr>
<tr>
<td></td>
<td><code>set currently active shader instance</code></td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::ShaderInstance&gt; &amp;</code></td>
<td><code>GetActiveShaderInstance() const</code></td>
</tr>
<tr>
<td></td>
<td><code>get currently active shader instance</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>ResetFeatureBits()</code></td>
</tr>
<tr>
<td></td>
<td><code>reset the current feature bits</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>SetFeatureBits(CoreGraphics::ShaderFeature::Mask m)</code></td>
</tr>
<tr>
<td></td>
<td><code>set shader feature by bit mask</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>ClearFeatureBits(CoreGraphics::ShaderFeature::Mask m)</code></td>
</tr>
<tr>
<td></td>
<td><code>clear shader feature by bit mask</code></td>
</tr>
<tr>
<td><code>CoreGraphics::ShaderFeature::Mask</code></td>
<td><code>GetFeatureBits() const</code></td>
</tr>
<tr>
<td></td>
<td><code>get the current feature mask</code></td>
</tr>
<tr>
<td><code>CoreGraphics::ShaderFeature::Mask</code></td>
<td><code> FeatureStringToMask(const Util::String &amp;str)</code></td>
</tr>
<tr>
<td></td>
<td><code>convert a shader feature string into a feature bit mask</code></td>
</tr>
<tr>
<td><code>Util::String</code></td>
<td><code>FeatureMaskToString(CoreGraphics::ShaderFeature::Mask m)</code></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>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) 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>
</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
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.

```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`
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  

Direct3D9::D3D9ShaderVariable
Direct3D9::D3D9ShaderVariable Class Reference

#include <d3d9shadervariable.h>

Inheritance diagram for Direct3D9::D3D9ShaderVariable:

```
Core::RefCounted

Base::ShaderVariableBase

Direct3D9::D3D9ShaderVariable

CoreGraphics::ShaderVariable
```
Detailed Description

D3D9 implementation of ShaderVariable.

(C) 2007 Radon Labs GmbH
Public Types

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>enum</strong></td>
<td><strong>Type</strong></td>
</tr>
<tr>
<td></td>
<td>shader variable types</td>
</tr>
<tr>
<td><strong>typedef</strong></td>
<td><strong>Name</strong></td>
</tr>
<tr>
<td><strong>Util::Atom</strong></td>
<td><strong>Semantic</strong></td>
</tr>
<tr>
<td><strong>&lt; Util::String &gt;</strong></td>
<td>shader variable semantic typedef</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9ShaderVariable ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~D3D9ShaderVariable ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>void SetInt (int value)</strong></td>
<td>Set int value</td>
</tr>
<tr>
<td>*<em>void SetIntArray (const int <em>values, SizeT count)</em></em></td>
<td>Set int array values</td>
</tr>
<tr>
<td><strong>void SetFloat (float value)</strong></td>
<td>Set float value</td>
</tr>
<tr>
<td>*<em>void SetFloatArray (const float <em>values, SizeT count)</em></em></td>
<td>Set float array values</td>
</tr>
<tr>
<td><strong>void SetVector (const Math::float4 &amp;value)</strong></td>
<td>Set vector value</td>
</tr>
<tr>
<td>*<em>void SetVectorArray (const Math::float4 <em>values, SizeT count)</em></em></td>
<td>Set vector array values</td>
</tr>
<tr>
<td><strong>void SetMatrix (const Math::matrix44 &amp;value)</strong></td>
<td>Set matrix value</td>
</tr>
<tr>
<td>*<em>void SetMatrixArray (const Math::matrix44 <em>values, SizeT count)</em></em></td>
<td>Set matrix array values</td>
</tr>
<tr>
<td><strong>void SetBool (bool value)</strong></td>
<td>Set bool value</td>
</tr>
<tr>
<td>*<em>void SetBoolArray (const bool <em>values, SizeT count)</em></em></td>
<td>Set bool array values</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetTexture</strong> (const Ptr&lt;CoreGraphics::Texture &amp;value)</td>
</tr>
<tr>
<td>Ptr</td>
<td><strong>CreateInstance</strong> ()</td>
</tr>
<tr>
<td>const Name &amp;</td>
<td><strong>GetName</strong> () const</td>
</tr>
<tr>
<td>const Semantic &amp;</td>
<td><strong>GetSemantic</strong> () const</td>
</tr>
<tr>
<td>SizeT</td>
<td><strong>GetNumArrayElements</strong> () const</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsArray</strong> () const</td>
</tr>
<tr>
<td>int</td>
<td><strong>GetRefCount</strong> () const</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</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><code>IsA</code> (const <code>Util::String</code> &amp;rttiName) 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</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA</code> (const <code>Util::FourCC</code> &amp;rttiFourCC) const</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 <code>Util::String</code> &amp;</td>
<td><code>GetClassName</code> () const</td>
</tr>
<tr>
<td></td>
<td><code>get the class name</code></td>
</tr>
<tr>
<td><code>Util::FourCC</code></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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static Util::String</code> <strong>ToString</strong> (Type t)</td>
<td>convert type to string</td>
</tr>
<tr>
<td><code>static void</code> <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 Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>setType</strong> <em>(Type t)</em></td>
<td>set variable type</td>
</tr>
<tr>
<td>void</td>
<td><strong>setName</strong> <em>(const Name &amp;n)</em></td>
<td>set variable name</td>
</tr>
<tr>
<td>void</td>
<td><strong>setSemantic</strong> <em>(const Semantic &amp;s)</em></td>
<td>set variable semantic</td>
</tr>
<tr>
<td>void</td>
<td><strong>setNumArrayElements</strong> <em>(SizeT n)</em></td>
<td>set number of array elements</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
## Public Member Functions

<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>const <strong>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 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, 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

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>
</table>
| `SetName`      | void (const Name &n)  
*set variation name* |
| `SetFeatureMask` | void (CoreGraphics::ShaderFeature::Mask m)  
*set feature bit mask of this variation* |
| `SetNumPasses` | void (SizeT n)  
*set number of passes* |
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::D3D9ShapeRenderer
Direct3D9::D3D9ShapeRenderer Class Reference

#include <d3d9shaperenderer.h>

Inheritance diagram for Direct3D9::D3D9ShapeRenderer:
Detailed Description

D3D9 implementation of ShapeRenderer.

(C) 2007 Radon Labs GmbH
Public Types

enum ShapeType

shape type
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D3D9ShapeRenderer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~D3D9ShapeRenderer ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void Open ()</td>
<td>open the shape renderer</td>
</tr>
<tr>
<td>void Close ()</td>
<td>close the shape renderer</td>
</tr>
<tr>
<td>void DrawShape (const Math::matrix44 &amp;modelTransform, ShapeType shapeType, const Math::float4 &amp;color)</td>
<td>draw a shape</td>
</tr>
<tr>
<td>void DrawPrimitives (const Math::matrix44 &amp;modelTransform, CoreGraphics::PrimitiveTopology::Code topology, SizeT numPrimitives, float *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, float *vertices, SizeT numVertices, SizeT vertexWidth, void *indices, CoreGraphics::IndexType::Code indexType, const Math::float4 &amp;color)</td>
<td>draw indexed primitives</td>
</tr>
<tr>
<td>bool isOpen () const</td>
<td>return true if open</td>
</tr>
<tr>
<td>void Begin ()</td>
<td>begin drawing shapes</td>
</tr>
<tr>
<td>void End ()</td>
<td>end drawing shapes</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>Function Type</td>
<td>Function Name</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>AddRef()</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>Release()</code></td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsInstanceOf(const Rtti &amp;rtti)</code> const</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsInstanceOf(const Util::String &amp;className)</code> const</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsInstanceOf(const Util::FourCC &amp;classFourCC)</code> const</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsA(const Rtti &amp;rtti)</code> const</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsA(const Util::String &amp;rttiName)</code> const</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>IsA(const Util::FourCC &amp;rttiFourCC)</code> const</td>
</tr>
<tr>
<td><code>const Util::String &amp;</code></td>
<td><code>GetClassName()</code> const</td>
</tr>
<tr>
<td><code>Util::FourCC</code></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>
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
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</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 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 void OnLoad Cancelled ()</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>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>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>
<tbody>
<tr>
<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

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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::D3D9StreamTextureLoader
Direct3D9::D3D9StreamTextureLoader
Class Reference

#include <d3d9streamtextureloader.h>

Inheritance diagram for Direct3D9::D3D9StreamTextureLoader:
Detailed Description

D3D9 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()</td>
<td>Return true if asynchronous loading is supported</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()</td>
<td>Return true if attached to resource</td>
</tr>
<tr>
<td>const Ptr&lt;Resource&gt;&amp; GetResource()</td>
<td>Get pointer to resource</td>
</tr>
<tr>
<td>Resource::State GetState()</td>
<td>Return current state</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</td>
<td><code>IsA</code> (const <code>Rtti</code> &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><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></th>
<th>DumpRefCountingLeaks ()</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>
Protected Member Functions

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

set current state
```
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::D3D9StreamTextureSaver
#include <d3d9streamtexturesaver.h>

Inheritance diagram for Direct3D9::D3D9StreamTextureSaver:
Detailed Description

D3D9 implementation of StreamTextureSaver.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual bool OnSave()</td>
<td>called by resource when a save is requested</td>
</tr>
<tr>
<td>void SetStream (const Ptr<a href="">IO::Stream</a> &amp;stream)</td>
<td>set stream to save to</td>
</tr>
<tr>
<td>const Ptr<a href="">IO::Stream</a> &amp; GetStream() const</td>
<td>get save-stream</td>
</tr>
<tr>
<td>void SetFormat (CoreGraphics::ImageFileFormat::Code fmt)</td>
<td>set file format (default is JPG)</td>
</tr>
<tr>
<td>CoreGraphics::ImageFileFormat::Code GetFormat() const</td>
<td>get file format</td>
</tr>
<tr>
<td>void SetMipLevel (IndexT mipLevel)</td>
<td>set the mip level to save (default is 0, the top level)</td>
</tr>
<tr>
<td>IndexT GetMipLevel() const</td>
<td>get the mip level to save</td>
</tr>
<tr>
<td>virtual void 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 OnRemoveFromResource ()</td>
<td>called when the resource saver 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>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</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>
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.
Direct3D9::D3D9Texture
Direct3D9::D3D9Texture Class Reference

#include <d3d9texture.h>

Inheritance diagram for Direct3D9::D3D9Texture:
Detailed Description

D3D9 implementation of Texture class.

FIXME: need to handle DeviceLost through RenderDevice event handler!

(C) 2007 Radon Labs GmbH
**Public Types**

<table>
<thead>
<tr>
<th>enum</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>texture types</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th>CubeFace</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cube map face</td>
</tr>
</tbody>
</table>

<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><strong>D3D9Texture ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~D3D9Texture ()</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><strong>Map (IndexT mipLevel, MapType mapType, MapInfo &amp;outMapInfo)</strong></td>
<td>map 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, MapType mapType, 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>GetD3D9BaseTexture ()</strong> const</td>
<td>get d3d9 base texture pointer</td>
</tr>
<tr>
<td><strong>GetD3D9Texture ()</strong> const</td>
<td>get d3d9 texture pointer</td>
</tr>
<tr>
<td><strong>GetD3D9CubeTexture ()</strong> const</td>
<td>get d3d9 cube texture pointer</td>
</tr>
<tr>
<td><strong>GetD3D9VolumeTexture ()</strong> const</td>
<td>get d3d9 volume texture pointer</td>
</tr>
<tr>
<td><strong>GetType ()</strong> const</td>
<td>get texture type</td>
</tr>
<tr>
<td><strong>GetWidth ()</strong> const</td>
<td>get width of texture</td>
</tr>
<tr>
<td><strong>GetHeight ()</strong> const</td>
<td>get height of texture (if 2d or 3d texture)</td>
</tr>
<tr>
<td><strong>GetDepth ()</strong> const</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetNumMipLevels()</code> const</td>
<td>get number of mip levels</td>
</tr>
<tr>
<td><code>GetPixelFormat()</code> const</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>GetUsage()</code> const</td>
<td>get resource usage type</td>
</tr>
<tr>
<td><code>GetAccess()</code> const</td>
<td>get cpu access type</td>
</tr>
<tr>
<td><code>SetAsyncEnabled(bool b)</code></td>
<td>request synchronous/asynchronous resource loading</td>
</tr>
<tr>
<td><code>IsAsyncEnabled()</code> const</td>
<td>return true if asynchronous resource loading is enabled</td>
</tr>
<tr>
<td><code>SetResourceId(const ResourceId &amp;id)</code></td>
<td>set the resource identifier</td>
</tr>
<tr>
<td><code>GetResourceId()</code> const</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()</code> const</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()</code> const</td>
<td>get optional resource saver</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</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>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>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>return true if this object is instance of given class, or a derived class, by fourcc</td>
</tr>
</tbody>
</table>
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><code>DumpRefCountingLeaks ()</code></th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetupFromD3D9Texture (IDirect3DTexture9 *ptr)</code></td>
<td>setup from a IDirect3DTexture9</td>
</tr>
<tr>
<td><code>void SetupFromD3D9CubeTexture (IDirect3DCubeTexture9 *ptr)</code></td>
<td>setup from a IDirect3DCubeTexture</td>
</tr>
<tr>
<td><code>void SetupFromD3D9VolumeTexture (IDirect3DVolumeTexture9 *ptr)</code></td>
<td>setup from a IDirect3DVolumeTexture</td>
</tr>
<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 SetUsage (Usage usage)</code></td>
<td>set 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>void SetState (State s)</code></td>
<td>set current state</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 ( * IDirect3DTexture9 tex2D ) [protected]

setup from a IDirect3DTexture9

Helper method to setup the texture object from a D3D9 2D texture.

```cpp
void Direct3D9::D3D9Texture::SetupFromD3D9CubeTexture ( * IDirect3DCubeTexture9 texCube ) [protected]

setup from a IDirect3DCubeTexture

Helper method to setup the texture object from a D3D9 cube texture.

```cpp
void Direct3D9::D3D9Texture::SetupFromD3D9VolumeTexture ( * IDirect3DVolumeTexture9 texVolume ) [protected]

setup from a IDirect3DVolumeTexture

Helper method to setup the texture object from a D3D9 volume texture.

```cpp
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.

```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
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.
Direct3D9::D3D9Types
Direct3D9::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>
<tbody>
<tr>
<td><code>AsD3D9PixelFormat(CoreGraphics::PixelFormat::Code p)</code></td>
<td>convert Nebula pixel format to D3D9 pixel format</td>
</tr>
<tr>
<td><code>AsNebulaPixelFormat(D3DFORMAT f)</code></td>
<td>convert Direct3D to Nebula pixel format</td>
</tr>
<tr>
<td><code>AsD3D9VertexDeclarationType(CoreGraphics::VertexComponent::Format)</code></td>
<td>convert vertex component type to D3D9 declaration type</td>
</tr>
<tr>
<td><code>AsD3D9VertexDeclarationUsage(CoreGraphics::VertexComponent::SemanticName n)</code></td>
<td>convert vertex component semantic name as D3D9 declaration usage</td>
</tr>
<tr>
<td><code>AsD3D9PrimitiveType(CoreGraphics::PrimitiveTopology::Code)</code></td>
<td>convert primitive topology to D3D</td>
</tr>
<tr>
<td><code>AsD3D9MultiSampleType(CoreGraphics::AntiAliasQuality::Code c)</code></td>
<td>convert antialias quality to D3D multisample type</td>
</tr>
<tr>
<td><code>AsD3DXImageFileFormat(CoreGraphics::ImageFileFormat::Code c)</code></td>
<td>convert image file format to D3DX file format</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](http://www.doxygen.org) at Tue Feb 19 12:16:46 2008
Direct3D9::D3D9VertexBuffer
Direct3D9::D3D9VertexBuffer Class Reference

#include <d3d9vertexbuffer.h>

Inheritance diagram for Direct3D9::D3D9VertexBuffer:
Detailed Description

D3D9 implementation of VertexBuffer.

(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>D3D9VertexBuffer ()</strong></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td>virtual <strong>~D3D9VertexBuffer ()</strong></td>
<td><em>destructor</em></td>
</tr>
<tr>
<td>virtual void <strong>Unload ()</strong></td>
<td><em>unload the resource, or cancel the pending load</em></td>
</tr>
<tr>
<td>void * <strong>Map</strong> (MapType mapType)</td>
<td><em>map the vertices for CPU access</em></td>
</tr>
<tr>
<td>void <strong>Unmap ()</strong></td>
<td><em>unmap the resource</em></td>
</tr>
<tr>
<td>IDirect3DVertexBuffer9 * <strong>GetD3D9VertexBuffer ()</strong> const</td>
<td><em>get pointer to d3d9 vertex buffer object</em></td>
</tr>
<tr>
<td>const Ptr&lt;br CoreGraphics::VertexLayout &gt; &amp; <strong>GetVertexLayout ()</strong> const</td>
<td><em>get the vertex layout</em></td>
</tr>
<tr>
<td>SizeT <strong>GetNumVertices ()</strong> const</td>
<td><em>get number of vertices in the buffer</em></td>
</tr>
<tr>
<td>Usage <strong>GetUsage ()</strong> const</td>
<td><em>get resource usage type</em></td>
</tr>
<tr>
<td>Access <strong>GetAccess ()</strong> const</td>
<td><em>get cpu access type</em></td>
</tr>
<tr>
<td>void <strong>SetAsyncEnabled</strong> (bool b)</td>
<td><em>request synchronous/asynchronous resource loading</em></td>
</tr>
<tr>
<td>bool <strong>IsAsyncEnabled ()</strong> const</td>
<td><em>return true if asynchronous resource loading is enabled</em></td>
</tr>
<tr>
<td>void <strong>SetResourceId</strong> (const ResourceId &amp;id)</td>
<td><em>set the resource identifier</em></td>
</tr>
<tr>
<td>const ResourceId &amp; <strong>GetResourceid ()</strong> const</td>
<td><em>get the resource identifier</em></td>
</tr>
<tr>
<td>void <strong>SetLoader</strong> (const Ptr&lt;br ResourceLoader &amp;loader)</td>
<td><em>set the resource identifier</em></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>GetLoader</td>
<td>constPtr to optional resource loader</td>
</tr>
<tr>
<td>GetSaver</td>
<td>constPtr to optional resource saver</td>
</tr>
<tr>
<td>GetUseCount</td>
<td>get current use count</td>
</tr>
<tr>
<td>Load</td>
<td>load the resource</td>
</tr>
<tr>
<td>GetState</td>
<td>get current state</td>
</tr>
<tr>
<td>IsLoaded</td>
<td>return true if current state is Loaded</td>
</tr>
<tr>
<td>IsPending</td>
<td>return true if current state is Pending</td>
</tr>
<tr>
<td>LoadFailed</td>
<td>return true if current state is Failed</td>
</tr>
<tr>
<td>Save</td>
<td>save the resource</td>
</tr>
<tr>
<td>GetRefCount</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</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>IsInstanceOf</td>
<td>return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>IsInstanceOf</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>&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!)
### Protected Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>SetVertexLayout (const <strong>Ptr</strong>: CoreGraphics::VertexLayout &amp;vertexLayout)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>set vertex layout (set by resource loader)</em></td>
</tr>
<tr>
<td>void</td>
<td>SetNumVertices (SizeT numVertices)</td>
</tr>
<tr>
<td></td>
<td><em>set number of vertices (set by resource loader)</em></td>
</tr>
<tr>
<td>void</td>
<td>SetUsage (Usage usage)</td>
</tr>
<tr>
<td></td>
<td><em>set resource usage type</em></td>
</tr>
<tr>
<td>void</td>
<td>SetAccess (Access access)</td>
</tr>
<tr>
<td></td>
<td><em>set resource cpu access type</em></td>
</tr>
<tr>
<td>void</td>
<td>SetState (State s)</td>
</tr>
<tr>
<td></td>
<td><em>set current state</em></td>
</tr>
<tr>
<td>void</td>
<td>IncrUseCount ()</td>
</tr>
<tr>
<td></td>
<td><em>increment use count</em></td>
</tr>
<tr>
<td>void</td>
<td>DecrUseCount ()</td>
</tr>
<tr>
<td></td>
<td><em>decrement use count</em></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.
Direct3D9::D3D9VertexLayout
Direct3D9::D3D9VertexLayout Class Reference

#include <d3d9vertexlayout.h>

Inheritance diagram for Direct3D9::D3D9VertexLayout:
Detailed Description

D3D9-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><strong>D3D9VertexLayout ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~D3D9VertexLayout ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Setup (const Util::Array<a href="">CoreGraphics::VertexComponent</a> &amp;c)</strong></td>
<td>setup the vertex layout</td>
</tr>
<tr>
<td>IDirect3DVertexDeclaration9 * <strong>GetD3D9VertexDeclaration ()</strong> const</td>
<td>get pointer to d3d9 vertex declaration</td>
</tr>
<tr>
<td><strong>IsValid ()</strong> const</td>
<td>return true if valid has been setup</td>
</tr>
<tr>
<td><strong>GetNumComponents ()</strong> const</td>
<td>get number of components</td>
</tr>
<tr>
<td><strong>GetComponentAt (IndexT i)</strong> const</td>
<td>get vertex component at index</td>
</tr>
<tr>
<td><strong>HasComponent (CoreGraphics::VertexComponent::SemanticName, IndexT semIndex)</strong> const</td>
<td>return true if vertex component exists</td>
</tr>
<tr>
<td><strong>FindComponent (CoreGraphics::VertexComponent::SemanticName, IndexT semIndex)</strong> const</td>
<td>get index of vertex component by semantics</td>
</tr>
<tr>
<td><strong>GetVertexByteSize ()</strong> const</td>
<td>get the vertex stride in number of bytes</td>
</tr>
<tr>
<td><strong>GetComponentByteOffset (CoreGraphics::VertexComponent::SemanticName, IndexT semIndex)</strong> const</td>
<td>get component offset from start of vertex</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>
</tbody>
</table>
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 Protected Member Functions**

<table>
<thead>
<tr>
<th>static</th>
<th>Util::String</th>
<th><strong>BuildSignature</strong> (const Util::Array&lt; CoreGraphics::VertexComponent &gt; &amp;c)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><em>get sharing signature for a set of vertex components</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

IndexT
Base::VertexLayoutBase::GetComponentByteOffset (CoreGraphics::VertexComponent::SemanticName
  IndexT
)

get component offset from start of vertex

Returns the component offset in bytes from the beginning of a vertex
to the start of the given vertex components. Returns InvalidIndex if the
vertex component doesn't exist!

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.
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><strong>FrameBatch ()</strong>&lt;br&gt;constructor</td>
<td></td>
</tr>
<tr>
<td>virtual <strong>~FrameBatch ()</strong>&lt;br&gt;destructor</td>
<td></td>
</tr>
<tr>
<td>void <strong>Discard ()</strong>&lt;br&gt;discard the frame batch</td>
<td></td>
</tr>
<tr>
<td>void <strong>Render ()</strong>&lt;br&gt;render the batch</td>
<td></td>
</tr>
<tr>
<td>void <strong>SetShader</strong>&lt;br&gt;(const Ptr<a href="">CoreGraphics::ShaderInstance</a> &amp;shd)&lt;br&gt;set batch shader</td>
<td></td>
</tr>
<tr>
<td>const Ptr<a href="">CoreGraphics::ShaderInstance</a> &amp; <strong>GetShader</strong> () const&lt;br&gt;get batch shader</td>
<td></td>
</tr>
<tr>
<td>void <strong>SetType</strong>&lt;br&gt;(CoreGraphics::BatchType::Code c)&lt;br&gt;set batch type</td>
<td></td>
</tr>
<tr>
<td>CoreGraphics::BatchType::Code <strong>GetType</strong> () const&lt;br&gt;get batch type</td>
<td></td>
</tr>
<tr>
<td>void <strong>SetNodeFilter</strong>&lt;br&gt;(Models::ModelNodeType::Code f)&lt;br&gt;set model node filter</td>
<td></td>
</tr>
<tr>
<td>Models::ModelNodeType::Code <strong>GetNodeFilter</strong> () const&lt;br&gt;get model node filter</td>
<td></td>
</tr>
<tr>
<td>void <strong>SetLightingMode</strong>&lt;br&gt;(LightingMode::Code c)&lt;br&gt;set lighting mode</td>
<td></td>
</tr>
<tr>
<td>LightingMode::Code <strong>GetLightingMode</strong> () const&lt;br&gt;get lighting mode</td>
<td></td>
</tr>
<tr>
<td>void <strong>SetSortingMode</strong>&lt;br&gt;(SortingMode::Code c)&lt;br&gt;set sorting mode</td>
<td></td>
</tr>
<tr>
<td>SortingMode::Code <strong>GetSortingMode</strong> () const&lt;br&gt;get sorting mode</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>void</td>
<td>SetShaderFeatures</td>
</tr>
<tr>
<td>CoreGraphics::ShaderFeature::Mask</td>
<td>GetShaderFeatures</td>
</tr>
<tr>
<td>void</td>
<td>AddVariable</td>
</tr>
<tr>
<td>SizeT</td>
<td>GetNumVariables</td>
</tr>
<tr>
<td>const Ptr<a href="">CoreGraphics::ShaderVariableInstance</a> &amp;</td>
<td>GetVariableByIndex</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</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf</td>
</tr>
<tr>
<td>bool</td>
<td>IsA</td>
</tr>
<tr>
<td>bool</td>
<td>IsA</td>
</tr>
<tr>
<td>bool</td>
<td>IsA</td>
</tr>
<tr>
<td>const</td>
<td>return true if this object is instance of given class derived class, by fourcc</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------</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></td>
<td><strong>GetClassFourCC</strong> () 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>
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::FramePass
Frame::FramePass Class Reference

#include <framepass.h>

Inheritance diagram for Frame::FramePass:

```
Core::RefCounted
  ↓
Frame::FramePass
```
Detailed Description

A frame pass encapsulates all 3d rendering to a render target, organized into FrameBatches.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FramePass ()</td>
<td>constructor</td>
</tr>
<tr>
<td>~FramePass ()</td>
<td>destructor</td>
</tr>
<tr>
<td>Discard ()</td>
<td>discard the frame pass</td>
</tr>
<tr>
<td>Render ()</td>
<td>render the pass</td>
</tr>
<tr>
<td>SetName (const Resources::ResourceId &amp;resId)</td>
<td>set the name of the frame pass</td>
</tr>
<tr>
<td>GetName () const</td>
<td>get the name of the frame pass</td>
</tr>
<tr>
<td>SetShader (const Ptr<a href="">CoreGraphics::ShaderInstance</a> &amp;shd)</td>
<td>set pass shader</td>
</tr>
<tr>
<td>GetShader () const</td>
<td>get pass shader</td>
</tr>
<tr>
<td>SetRenderTarget (const Ptr<a href="">CoreGraphics::RenderTarget</a> &amp;rt)</td>
<td>set render target</td>
</tr>
<tr>
<td>GetRenderTarget () const</td>
<td>get render target</td>
</tr>
<tr>
<td>AddVariable (const Ptr<a href="">CoreGraphics::ShaderVariableInstance</a> &amp;var)</td>
<td>add a shader variable instance to the frame pass</td>
</tr>
<tr>
<td>GetNumVariables () const</td>
<td>get number of shader variables</td>
</tr>
<tr>
<td>GetVariableByIndex (IndexT i) const</td>
<td>get shader variable by index</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>void AddBatch (const Ptr&lt; FrameBatch &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;)</td>
<td>set clear color</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</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</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) 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 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 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>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</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 () 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.

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.
Frame::FramePostEffect
Frame::FramePostEffect Class Reference

#include <frameposteffect.h>

Inheritance diagram for Frame::FramePostEffect:

```
Core::RefCounted
  Frame::FramePostEffect
```
Detailed Description

A frame post-effect implements draws a fullscreen quad through a shader which implements the post effect.

(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><code>virtual ~FramePostEffect ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void Setup ()</code></td>
<td>setup the post effect</td>
</tr>
<tr>
<td><code>void Discard ()</code></td>
<td>discard the post effect</td>
</tr>
<tr>
<td><code>void Render ()</code></td>
<td>render the post effect</td>
</tr>
<tr>
<td><code>void SetName (const Resources::ResourceId &amp;resId)</code></td>
<td>set the name of the frame pass</td>
</tr>
<tr>
<td><code>const Resources::ResourceId &amp; GetName () const</code></td>
<td>get the name of the frame pass</td>
</tr>
<tr>
<td><code>void SetShader (const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp;shd)</code></td>
<td>set pass shader</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::ShaderInstance &gt; &amp; GetShader () const</code></td>
<td>get pass shader</td>
</tr>
<tr>
<td><code>void SetRenderTarget (const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp;rt)</code></td>
<td>set render target</td>
</tr>
<tr>
<td><code>const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; GetRenderTarget () const</code></td>
<td>get render target</td>
</tr>
<tr>
<td><code>void AddVariable (const Ptr&lt; CoreGraphics::ShaderVariableInstance &gt; &amp;var)</code></td>
<td>add a shader variable instance to the frame pass</td>
</tr>
<tr>
<td><code>SizeT GetNumVariables () const</code></td>
<td>get number of shader variables</td>
</tr>
<tr>
<td><code>const Ptr&lt; GetVariableByIndex (IndexT i) const</code></td>
<td>get shader variable by index</td>
</tr>
<tr>
<td><strong>&lt; CoreGraphics::ShaderVariableInstance &gt;</strong></td>
<td>get shader variable by index</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 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 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 Frame::FramePostEffect::Setup()

setup the post effect

This setup the quad mesh for rendering the fullscreen quad.

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

Frame::FrameServer
Frame::FrameServer Class Reference

#include <frameserver.h>

Inheritance diagram for Frame::FrameServer:

```
 Core::RefCounted
   
 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><strong>FrameServer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~FrameServer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>bool Open ()</strong></td>
<td>Open the frame server (loads all frame shaders)</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>Close the frame server</td>
</tr>
<tr>
<td><strong>bool IsOpen () const</strong></td>
<td>Return true if open</td>
</tr>
<tr>
<td><strong>bool HasFrameShader (const Resources::ResourceId &amp;name) const</strong></td>
<td>Return true if a frame shader exists</td>
</tr>
<tr>
<td><strong>const Ptr &lt; FrameShader &gt; &amp; GetFrameShaderByName (const Resources::ResourceId &amp;name) const</strong></td>
<td>Get frame shader by name</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</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</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 string</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>
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::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>FrameShader ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~FrameShader ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void Discard ()</td>
<td>discard the frame shader</td>
</tr>
<tr>
<td>void Render ()</td>
<td>render the frame shader from the given camera</td>
</tr>
<tr>
<td>void SetName (const Resources::ResourceId &amp;id)</td>
<td>set the name of the frame shader</td>
</tr>
<tr>
<td>const Resources::ResourceId &amp; GetName ()</td>
<td>get the name of the frame shader</td>
</tr>
<tr>
<td>void SetMainRenderTarget (const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp;rt)</td>
<td>set the main render target</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; GetMainRenderTarget ()</td>
<td>get the main render target</td>
</tr>
<tr>
<td>void AddRenderTarget (const Resources::ResourceId &amp;resId, const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp;rt)</td>
<td>add a render target to the frame</td>
</tr>
<tr>
<td>SizeT GetNumRenderTargets ()</td>
<td>get number of render targets</td>
</tr>
<tr>
<td>const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; GetRenderTargetByIndex (IndexT i)</td>
<td>get render target by index</td>
</tr>
<tr>
<td>bool HasRenderTarget (const Resources::ResourceId &amp;resId)</td>
<td>return true if render target exists by name</td>
</tr>
<tr>
<td>const Ptr GetRenderTargetByName</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>get render target by name</code></td>
<td><code>get render target by name</code></td>
</tr>
</tbody>
</table>
| `Add a frame pass to the frame shader` | `AddFramePass` (const Ptr<FramePass> &framePass)

SizeT `GetNumFramePasses () const` | get number of frame passes |
| `get frame pass by index` | `GetFramePassByIndex` (IndexT i) const |
| `return true if names pass exists` | `HasFramePass` (const Resources::ResourceId &resId) const |
| `get frame pass by name` | `GetFramePassByName` (const Resources::ResourceId &resId) const |
| `add a post effect to the frame shader` | `AddPostEffect` (constPtr<FramePostEffect> &postEffect) |
| `get number of post effects` | `GetNumPostEffects () const` |
| `get post effect by index` | `GetPostEffectByIndex` (IndexT i) const |
| `return true if post effect exists` | `HasPostEffect` (const Resources::ResourceId &resId) const |
| `get post effect by name` | `GetPostEffectByName` (const Resources::ResourceId &resId) const |
| `get the current refcount` | `GetRefCount () const` |
| `increment refcount by one` | `AddRef ()` |
| `decrement refcount and destroy object if refcount is` | `Release ()` |
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
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

```cpp
static Util::Array< Ptr< FrameShader > > LoadFrameShaders (const IO::URI &uri)
```

*load a frame shaders from an XML file*

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:46 2008
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><em>lighting mode enum</em></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>static <code>Code</code></td>
<td><code>FromString (const Util::String &amp;str)</code></td>
<td>convert from string</td>
</tr>
<tr>
<td>static <code>Util::String</code></td>
<td><code>ToString (Code c)</code></td>
<td>convert to string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:46 2008
- 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>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sorting mode enum</td>
</tr>
<tr>
<td>static Code</td>
<td>Function</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>static</td>
<td><code>FromString</code> (const Util::String &amp;str)</td>
</tr>
<tr>
<td>static Util::String</td>
<td><code>ToString</code> (Code c)</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Tue Feb 19 12:16:46 2008
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 specilized entitymanager), derive from this class and overwrite OnActivate() OnDeactivate().

(C) 2007 Radon Labs GmbH
Game::CoreFeature
Game::CoreFeature Class Reference

#include <corefeatureunit.h>
Detailed Description

The CoreFeature just creates all core systems of Nebula3. That are Core, Io, Script and the Http subsystems.

(C) 2007 Radon Labs GmbH
Game::Entity
#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

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>typedef unsigned int EntityId</code></td>
<td>an Id type, used to identify entities</td>
</tr>
</tbody>
</table>
## 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>virtual <strong>~Entity ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>const <strong>Util::String &amp; GetCategory () const</strong></td>
<td>get the entity’s category</td>
</tr>
<tr>
<td><strong>EntityId GetUniqueld () const</strong></td>
<td>get unique id of entity</td>
</tr>
<tr>
<td>bool <strong>AcceptsMessage (const Messaging::Id &amp;msgId) const</strong></td>
<td>return true if any property accepts/processes message</td>
</tr>
<tr>
<td>void <strong>SendSync (const Ptr&lt;Messaging::Message &amp;&gt; &amp;msg)</strong></td>
<td>send a synchronous message to the entity</td>
</tr>
<tr>
<td>bool <strong>IsActive () const</strong></td>
<td>return true if the entity is currently active (between OnActivate/OnDeactivate)</td>
</tr>
<tr>
<td>const <strong>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>void <strong>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>void <strong>OnActivate ()</strong></td>
<td>called when attached to world</td>
</tr>
<tr>
<td>void <strong>OnDeactivate ()</strong></td>
<td>called when removed from world</td>
</tr>
<tr>
<td>void <strong>OnBeginFrame ()</strong></td>
<td>called at the beginning of the frame</td>
</tr>
</tbody>
</table>
void **OnMoveBefore** ()
called before movement

void **OnMoveAfter** ()
called after movement

void **OnRender** ()
called before rendering

void **OnRenderDebug** ()
debug rendering called before rendering

void **OnLoseActivity** ()
called if entity loses activity

void **OnGainActivity** ()
called if entity gains activity

void **OnLoad** ()
called after loading from database has happened

void **OnStart** ()
called when the entity starts to live in the complete world

void **OnSave** ()
called before saving from database happens

bool **HasAttr** (const **Attr::AttrId** &attrId) const
return true if entity has an attribute

void **SetAttr** (const **Attr::Attribute** &attr)
generic attribute setter (slow!)

**Attr::Attribute** **GetAttr** (const **Attr::AttrId** &attrId) const
generic attribute getter (slow!)

void **SetString** (const **Attr::StringAttrId** &attrId, const **Util::String** &s)
set string attribute on the entity

const **Util::String** & **GetString** (const **Attr::StringAttrId** &attrId) const
get string attribute from the entity

void **SetInt** (const **Attr::IntAttrId** &attrId, int i)
set int attribute on the entity

int **GetInt** (const **Attr::IntAttrId** &attrId) const
get int attribute from the entity

void **SetFloat** (const **Attr::FloatAttrId** &attrId, float f)
set float attribute on the entity

float **GetFloat** (const **Attr::FloatAttrId** &attrId) const
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetBool (const Attr::BoolAttrId &amp;attrId, bool b)</code></td>
<td>set bool attribute on the entity</td>
</tr>
<tr>
<td><code>bool GetBool (const Attr::BoolAttrId &amp;attrId)</code></td>
<td>get bool attribute from the entity</td>
</tr>
<tr>
<td><code>void SetFloat4 (const Attr::Float4AttrId &amp;attrId, const Math::float4 &amp;v)</code></td>
<td>set float4 attribute on the entity</td>
</tr>
<tr>
<td><code>const Math::float4 GetFloat4 (const Attr::Float4AttrId &amp;attrId)</code></td>
<td>get float4 attribute from the entity</td>
</tr>
<tr>
<td><code>void SetMatrix44 (const Attr::Matrix44AttrId &amp;attrId, const Math::matrix44 &amp;m)</code></td>
<td>set matrix44 attribute on the entity</td>
</tr>
<tr>
<td><code>const Math::matrix44 GetMatrix44 (const Attr::Matrix44AttrId &amp;attrId)</code></td>
<td>get matrix44 attribute from the entity</td>
</tr>
<tr>
<td><code>void SetGuid (const Attr::GuidAttrId &amp;attrId, const Util::Guid &amp;guid)</code></td>
<td>set guid attribute on the entity</td>
</tr>
<tr>
<td><code>const Util::Guid &amp; GetGuid (const Attr::GuidAttrId &amp;attrId)</code></td>
<td>get guid attribute from the entity</td>
</tr>
<tr>
<td><code>void SetBlob (const Attr::BlobAttrId &amp;attrId, const Util::Blob &amp;blob)</code></td>
<td>set blob attribute on the entity</td>
</tr>
<tr>
<td><code>const Util::Blob &amp; GetBlob (const Attr::BlobAttrId &amp;attrId)</code></td>
<td>get blob attribute from the entity</td>
</tr>
<tr>
<td><code>void AddString (const Attr::StringAttrId &amp;attrId)</code></td>
<td>add string attribute if not exists</td>
</tr>
<tr>
<td><code>void AddInt (const Attr::IntAttrId &amp;attrId)</code></td>
<td>add int attribute if not exists</td>
</tr>
<tr>
<td><code>void AddFloat (const Attr::FloatAttrId &amp;attrId)</code></td>
<td>add float attribute if not exists</td>
</tr>
<tr>
<td><code>void AddBool (const Attr::BoolAttrId &amp;attrId)</code></td>
<td>add bool attribute if not exists</td>
</tr>
<tr>
<td><code>void AddFloat4 (const Attr::Float4AttrId &amp;attrId)</code></td>
<td>add float4 attribute if not exists</td>
</tr>
<tr>
<td><code>void AddMatrix44 (const Attr::Matrix44AttrId &amp;attrId)</code></td>
<td>add matrix44 attribute if not exists</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>AddGuid</code></td>
<td>Add guid attribute if not exists</td>
</tr>
<tr>
<td><code>AddBlob</code></td>
<td>Add blob attribute if not exists</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><code>IsInstanceOf</code></td>
<td>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>Return true if this object is instance of given class, by string</td>
</tr>
<tr>
<td><code>IsA</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

<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
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
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.

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.

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.
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 const
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
Game::FeatureUnit Class Reference

#include <featureunit.h>

Inheritance diagram for Game::FeatureUnit:

```
Core::RefCounted

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><strong>FeatureUnit ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~FeatureUnit ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual OnActivate ()</strong></td>
<td>called from GameServer::ActivateProperties()</td>
</tr>
<tr>
<td><strong>virtual OnDeactivate ()</strong></td>
<td>called from GameServer::DeactivateProperties()</td>
</tr>
<tr>
<td><strong>bool IsActive () const</strong></td>
<td>return true if property is currently active</td>
</tr>
<tr>
<td><strong>virtual OnLoad ()</strong></td>
<td>called from within GameServer::Load() after attributes are loaded</td>
</tr>
<tr>
<td><strong>virtual OnStart ()</strong></td>
<td>called from within GameServer::OnStart() after OnLoad when the complete world exist</td>
</tr>
<tr>
<td><strong>virtual OnSave ()</strong></td>
<td>called from within GameServer::Save() before attributes are saved back to database</td>
</tr>
<tr>
<td><strong>virtual OnBeginFrame ()</strong></td>
<td>called on begin of frame</td>
</tr>
<tr>
<td><strong>virtual OnFrame ()</strong></td>
<td>called in the middle of the feature trigger cycle</td>
</tr>
<tr>
<td><strong>virtual OnEndFrame ()</strong></td>
<td>called at the end of the feature trigger cycle</td>
</tr>
<tr>
<td><strong>virtual OnRenderDebug ()</strong></td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td><strong>virtual AttachManager (const Ptr&lt; Manager &gt; &amp;manager)</strong></td>
<td>attach a manager to the game world</td>
</tr>
<tr>
<td><strong>virtual RemoveManager (const Ptr&lt; Manager &gt; &amp;manager)</strong></td>
<td>remove a manager from the game world</td>
</tr>
<tr>
<td><strong>void SetCmdLineArgs (const Util::CmdLineArgs &amp;a)</strong></td>
<td>set command line args</td>
</tr>
<tr>
<td><strong>int GetRefCount () const</strong></td>
<td></td>
</tr>
</tbody>
</table>
get the current refcount

```cpp
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
```

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

void Game::FeatureUnit::OnActivate() [virtual]
called from GameServer::ActivateProperties()

This method is called by Game::GameServer::ActivateProperties().
Use this method for one-time initializations of the FeatureUnit.

void Game::FeatureUnit::OnDeactivate() [virtual]
called from GameServer::DeactivateProperties()

This method is called by Game::GameServer::DeactivateProperties().
Use this method to cleanup stuff which has been initialized in OnActivate().

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.

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.

```cpp
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.

```cpp
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.

```cpp
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.

```cpp
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 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.
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><code>GameServer()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~GameServer()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual bool <code>Open()</code></td>
<td>open the game world</td>
</tr>
<tr>
<td>virtual void <code>Close()</code></td>
<td>close the game world</td>
</tr>
<tr>
<td>virtual bool <code>Start()</code></td>
<td>start the game world</td>
</tr>
<tr>
<td><code>HasStarted()</code></td>
<td>has the game world already started</td>
</tr>
<tr>
<td>virtual void <code>Stop()</code></td>
<td>stop the game world</td>
</tr>
<tr>
<td>virtual void <code>OnFrame()</code></td>
<td>trigger the game world</td>
</tr>
<tr>
<td>virtual void <code>NotifyGameLoad()</code></td>
<td>call <code>OnLoad</code> on all game features</td>
</tr>
<tr>
<td>virtual void <code>NotifyGameSave()</code></td>
<td>call <code>OnSave</code> on all game features</td>
</tr>
<tr>
<td>void <code>AttachGameFeature(const Ptr&lt;FeatureUnit&gt;&amp; feature)</code></td>
<td>add game feature</td>
</tr>
<tr>
<td>void <code>RemoveGameFeature(const Ptr&lt;FeatureUnit&gt;&amp; feature)</code></td>
<td>remove game feature</td>
</tr>
<tr>
<td>bool <code>IsQuitRequested()</code></td>
<td>is quit requested</td>
</tr>
<tr>
<td>void <code>RequestQuit()</code></td>
<td>request quit</td>
</tr>
<tr>
<td>int <code>GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void <code>AddRef()</code></td>
<td>add reference count</td>
</tr>
</tbody>
</table>
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><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
Protected Member Functions

void RenderDebug ()
 render a debug visualization
Member Function Documentation

bool
Game::GameServer::Open() [virtual]

open the game world

Initialize the game server object. This will create and initialize all subsystems.

void
Game::GameServer::Close() [virtual]

close the game world

Close the game server object.

bool
Game::GameServer::Start() [virtual]

start the game world

Start the game world, called after loading has completed.

void
Game::GameServer::Stop() [virtual]

stop the game world

Stop the game world, called before the world(current level) is cleaned up.

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.
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>Function</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 Game::Server::Start() 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></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</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>GetSizeT GetNumHandlers()</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()</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()</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>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</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>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

```cpp
void RegisterMessage (const Id &msgId)

register a single accepted message
```
Member Function Documentation

```cpp
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::CategoryManager`, `BaseGameFeature::EnvQueryManager`, and `BaseGameFeature::GlobalAttrsManager`.

```cpp
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::CategoryManager`, `BaseGameFeature::EntityManager`, `BaseGameFeature::EnvEntityManager`, and `BaseGameFeature::EnvQueryManager`.

```cpp
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.

```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
  const
  ```
```cpp
bool Messaging::Dispatcher::HasPort(const Messaging::Port & port) const [inherited]
return true if a port exists
Return true if a port is already attached.

void Messaging::Port::AttachHandler(const Messaging::Handler & h) [inherited]
attach a message handler to the port
Attach a message handler to the port.

void Messaging::Port::RemoveHandler(const Messaging::Handler & h) [inherited]
remove a message handler from the port
Remove a message handler from the port.

void Messaging::Port::Send(const Messaging::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 actor physics 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
Game::Property
#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

<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><strong>Property()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~Property()</strong></td>
<td>Destructor</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>SetupDefaultAttributes()</strong></td>
<td>Setup the property's attributes to their default state</td>
</tr>
<tr>
<td><strong>SetupCallbacks()</strong></td>
<td>Setup callbacks for this property, call by entity in <strong>OnActivate()</strong></td>
</tr>
<tr>
<td><strong>OnActivate()</strong></td>
<td>Called from <strong>Entity::ActivateProperties()</strong></td>
</tr>
<tr>
<td><strong>OnDeactivate()</strong></td>
<td>Called from <strong>Entity::DeactivateProperties()</strong></td>
</tr>
<tr>
<td><strong>IsActive() const</strong></td>
<td>Return true if property is currently active</td>
</tr>
<tr>
<td><strong>OnLoad()</strong></td>
<td>Called from within <strong>Entity::Load()</strong> after attributes are loaded</td>
</tr>
<tr>
<td><strong>OnStart()</strong></td>
<td>Called from within <strong>Entity::OnStart()</strong> after <strong>OnLoad</strong> when the complete world exist</td>
</tr>
<tr>
<td><strong>OnSave()</strong></td>
<td>Called from within <strong>Entity::Save()</strong> 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>OnRender()</strong></td>
<td>Called before rendering happens</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>virtual void <strong>OnRenderDebug</strong> ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void <strong>OnLoseActivity</strong> ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void <strong>OnGainActivity</strong> ()</td>
<td>called when game debug visualization is on</td>
</tr>
<tr>
<td>virtual void <strong>HandleMessage</strong> (const Ptr&lt; Messaging::Message &gt; &amp;msg)</td>
<td>handle a single message</td>
</tr>
<tr>
<td>virtual void <strong>SetupAcceptedMessages</strong> ()</td>
<td>override to register accepted messages</td>
</tr>
<tr>
<td>void <strong>AttachHandler</strong> (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>attach a message handler to the port</td>
</tr>
<tr>
<td>void <strong>RemoveHandler</strong> (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>remove a message handler from the port</td>
</tr>
<tr>
<td>SizeT <strong>GetNumHandlers</strong> () const</td>
<td>return number of handlers attached to the port</td>
</tr>
<tr>
<td>const Ptr&lt; Handler &gt; &amp; <strong>GetHandlerAtIndex</strong> (IndexT i) const</td>
<td>get a message handler by index</td>
</tr>
<tr>
<td>virtual void <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; <strong>GetAcceptedMessages</strong> () const</td>
<td>get the array of accepted messages (sorted)</td>
</tr>
<tr>
<td>bool <strong>AcceptsMessage</strong> (const Id &amp;msgId) const</td>
<td>return true if port accepts this msg</td>
</tr>
<tr>
<td>virtual void <strong>HandleMessage</strong> (const Ptr&lt; Messaging::Message &gt; &amp;msg)</td>
<td>handle a single accepted message</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 <strong>Util::String</strong> &amp;className) 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><strong>IsInstanceOf</strong> (const <strong>Util::FourCC</strong> &amp;classFourCC) 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><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</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>
<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><code>void SetEntity (const Ptr&lt; Entity &gt; &amp;v)</code></td>
<td>Sets the entity this is attached to to <code>v</code>.</td>
</tr>
<tr>
<td><code>void ClearEntity ()</code></td>
<td>Removes the entity.</td>
</tr>
<tr>
<td><code>void RegisterMessage (const Id &amp;msgId)</code></td>
<td>Registers a single accepted message.</td>
</tr>
</tbody>
</table>
Member Function Documentation

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.

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.

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.

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().
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.

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.

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).

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.

```cpp
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.

```cpp
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.

```cpp
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.

```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.
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).

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).

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.

```cpp
void Messaging::Port::HandleMessage(const Ptr<Messaging::Message> &msg) [virtual, inherited]
```

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().

Reimplemented in Messaging::Dispatcher.

```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

Graphics::ActorEntity
Graphics::ActorEntity Class Reference

#include <actorentity.h>

Inheritance diagram for Graphics::ActorEntity:
Detailed Description

An actor graphics entity. NOTE: The current ActorEntity class is just a wrapper around the legacy Nebula2 Character3 system and will be replaced with a "proper" Nebula3 character system in the future.

(C) 2007 Radon Labs GmbH
**Public Types**

<table>
<thead>
<tr>
<th>enum</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>entity types</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th>LinkType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>visibility link types</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ActorEntity ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~ActorEntity ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual void OnActivate ()</strong></td>
<td>called when attached to game entity</td>
</tr>
<tr>
<td><strong>virtual void OnDeactivate ()</strong></td>
<td>called when removed from game entity</td>
</tr>
<tr>
<td><strong>void SetAnimationMapping (const Util::String &amp;n)</strong></td>
<td>set animation mapping to use</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetAnimationMapping () const</strong></td>
<td>get animation mapping to use</td>
</tr>
<tr>
<td><strong>void SetSkinList (const Util::String &amp;s)</strong></td>
<td>set optional skin lists for character3's</td>
</tr>
<tr>
<td><strong>const Util::String &amp; GetSkinList () const</strong></td>
<td>get skin list</td>
</tr>
<tr>
<td><strong>void SetBaseAnimation (const Util::String &amp;anim, Timing::Time fadeIn=0.0, Timing::Time timeOffset=0.0, bool onlyIfInactive=true, bool waitForOverlayAnim=false, float randomValue=0.0f)</strong></td>
<td>set the current base animation (usually loop anims) - OBSOLETE, use FadeAnimation(), ClearAnimations()</td>
</tr>
<tr>
<td><strong>Util::String GetBaseAnimation () const</strong></td>
<td>get current base animation (first of mix)</td>
</tr>
<tr>
<td><strong>Util::String GetBaseClip () const</strong></td>
<td>get current base clip (first of mix)</td>
</tr>
<tr>
<td><strong>Timing::Time GetBaseAnimationDuration () const</strong></td>
<td>get base animation duration</td>
</tr>
<tr>
<td><strong>void SetOverlayAnimation (const Util::String &amp;anim, Timing::Time fadeIn=0.0, Timing::Time overrideDuration=0.0, bool</strong></td>
<td>set the current overlay animation (usually loop anims) - OBSOLETE, use FadeAnimation(), ClearAnimations()</td>
</tr>
</tbody>
</table>
onlyIfInactive=true, float randomValue=0.0f)

set the current overlay animation (usually oneshot anims)
- OBSOLETE, use `FadeAnimation()`, `ClearAnimations()`

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Util::String</td>
<td><code>GetOverlayAnimation()</code> const</td>
<td>get name of currently active overlay animation (first of mix)</td>
</tr>
<tr>
<td>Util::String</td>
<td><code>GetOverlayClip()</code> const</td>
<td>get name of currently active overlay clip (first of mix)</td>
</tr>
<tr>
<td>Timing::Time</td>
<td><code>GetOverlayAnimationDuration()</code> const</td>
<td>get overlay animation duration</td>
</tr>
<tr>
<td></td>
<td><code>SetFadeAnimationMix</code> (const <code>Util::Array&lt;Util::String&gt;</code> &amp;anims, const <code>Util::Array&lt;float&gt;</code> &amp;weights, <code>Timing::Time</code> fadeIn, <code>Timing::Time</code> overrideDuration, float randomValue)</td>
<td>set the current fade animations - OBSOLETE, use <code>FadeAnimation()</code>, <code>ClearAnimations()</code></td>
</tr>
<tr>
<td></td>
<td><code>StopOverlayAnimation</code> (<code>Timing::Time</code> fadeIn)</td>
<td>stop current overlay animation</td>
</tr>
<tr>
<td></td>
<td><code>IsOverlayAnimationActive()</code> const</td>
<td>return true if overlay animation is currently active</td>
</tr>
<tr>
<td></td>
<td><code>OnUpdate()</code></td>
<td>called before rendering happens</td>
</tr>
<tr>
<td>Char::Character *</td>
<td><code>GetCharacterPointer()</code> const</td>
<td>get pointer to our nCharacter2 object</td>
</tr>
<tr>
<td></td>
<td><code>EvaluateSkeleton</code> (bool enforceEvaluation=false)</td>
<td>bring the character's skeleton uptodate</td>
</tr>
<tr>
<td></td>
<td><code>HasCharacter3Set()</code> const</td>
<td>is this a character3 ?</td>
</tr>
<tr>
<td>Char::CharacterSet *</td>
<td><code>GetCharacter3Set()</code> const</td>
<td>get the character3 set, fail if no character3 set exists</td>
</tr>
<tr>
<td></td>
<td><code>GetJointIndexByName</code> (const <code>Util::String</code> &amp;name)</td>
<td>return joint index by name, return <code>InvalidIndex</code> if joint doesn't exist</td>
</tr>
<tr>
<td>Char::CharJoint *</td>
<td><code>GetJoint</code> (int jointIndex) 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><code>GetLocalJointMatrix(int jointIndex)</code></td>
<td>return a joint's current matrix in model space</td>
<td></td>
</tr>
<tr>
<td><code>GetJointMatrixByIndex(IndexT jointIndex)</code></td>
<td>get joint matrix by index in global space</td>
<td></td>
</tr>
<tr>
<td><code>GetJointMatrixByName(const Util::String &amp;jointName)</code></td>
<td>get joint matrix by name in global space, throws assert if joint doesn't exist</td>
<td></td>
</tr>
<tr>
<td><code>GetAnimClipScheduler()</code></td>
<td>return the animation clip scheduler</td>
<td></td>
</tr>
<tr>
<td><code>FadeAnimation(const Util::String &amp;anim, float targetWeight=0.0f, Timing::Time fadeInTime=0.0, Timing::Time fadeOutTime=0.0, float randomValue=0.0f, float runTime=0.0f, float timeFactor=1.0f, Timing::Time sampleTime=0.0, bool animRestart=false, bool fadeOutRunningAnims=true)</code></td>
<td>fade an animation to a weight</td>
<td></td>
</tr>
<tr>
<td><code>FadeAnimationMix(const Util::Array&lt;Util::String&gt; &amp;anims, float targetWeight=0.0f, Timing::Time fadeInTime=0.0, Timing::Time fadeOutTime=0.0, float randomValue=0.0f, float runTime=0.0f, float timeFactor=1.0f, Timing::Time sampleTime=0.0, bool animRestart=false, bool fadeOutRunningAnims=true)</code></td>
<td>fade a mix of animations to a weight</td>
<td></td>
</tr>
<tr>
<td><code>FadeAnimationMix(const Util::Array&lt;Util::String&gt; &amp;anims, const Util::Array&lt;float&gt; &amp;targetWeights, Timing::Time fadeInTime=0.0, Timing::Time fadeOutTime=0.0, float randomValue=0.0f, float runTime=0.0f, float timeFactor=1.0f, Timing::Time sampleTime=0.0, bool fadeOutRunningAnims=true)</code></td>
<td>fade a mix of animations to a weight</td>
<td></td>
</tr>
</tbody>
</table>
animRestart=false, bool
dfaeOutRunningAnims=true)
fade a mix of animations to various weights

void **FadeRunningAnimationsOut ()**
fade all running animations out

void **FadeRunningAnimationsOut** (AnimationType type, const **Util::Array**< int > *clipFilter=0)
fades running clips of the given type out

const **Ptr** < Models::CharacterNode > & **GetCharacterNode ()** const
get character node

void **SetResourceId** (const **Resources::ResourceId** &resId)
set the model's resource id

const **Resources::ResourceId** & **GetResourceId ()** const
get the model's resource id

**Resources::Resource::State** **GetModelResourceState ()** const
get the state of the contained managed model resource

const **Ptr** < Models::ModelInstance > & **GetModellInstance ()** const
get pointer to model instance (only valid if already loaded)

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 for rendering)

**Type** **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** < Stage > &
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetStage</strong> () const</td>
<td>Get the stage this entity is attached to</td>
</tr>
<tr>
<td><strong>IsAttachedToStage</strong> () const</td>
<td>Return true if entity is attached to stage</td>
</tr>
<tr>
<td>const <strong>Ptr</strong>&lt; <strong>Cell</strong> &gt; &amp; <strong>GetCell</strong> () const</td>
<td>Get the cell this entity is attached to</td>
</tr>
<tr>
<td>bool <strong>IsAttachedToCell</strong> () const</td>
<td>Return true if entity is attached to cell</td>
</tr>
<tr>
<td>const <strong>Math::bbox</strong> &amp; <strong>GetLocalBoundingBox</strong> () const</td>
<td>Get the local space bounding box</td>
</tr>
<tr>
<td>const <strong>Math::bbox</strong> &amp; <strong>GetGlobalBoundingBox</strong> () const</td>
<td>Get bounding box in global space</td>
</tr>
<tr>
<td>void <strong>ClearLinks</strong> (<strong>LinkType</strong> linkType)</td>
<td>Clear all visibility links</td>
</tr>
<tr>
<td>void <strong>AddLink</strong> (<strong>LinkType</strong> linkType, const <strong>Ptr</strong>&lt; <strong>GraphicsEntity</strong> &gt; &amp;entity)</td>
<td>Add visibility link</td>
</tr>
<tr>
<td>const <strong>Util::Array</strong>&lt; <strong>Ptr</strong>&lt; <strong>GraphicsEntity</strong> &gt; &gt; &amp; <strong>GetLinks</strong> (<strong>LinkType</strong> type) const</td>
<td>Get visibility links by type</td>
</tr>
<tr>
<td>virtual <strong>ComputeClipStatus</strong>::&lt; <strong>Math::ClipStatus::Type</strong>&gt; (const <strong>Math::bbox</strong> &amp;box)</td>
<td>Compute clip status against bounding box</td>
</tr>
<tr>
<td>void <strong>SetTime</strong> (<strong>Timing::Time</strong> t)</td>
<td>Set graphics time</td>
</tr>
<tr>
<td><strong>Timing::Time</strong> <strong>GetTime</strong> () const</td>
<td>Get graphics time</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;str) 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>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 string</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; &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>DumpRefCountingLeaks ()</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 OnTransformChanged ()</td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td>virtual void OnRenderDebug ()</td>
<td>called to render a debug visualization of the entity</td>
</tr>
<tr>
<td>void ValidateModelInstance ()</td>
<td>validate the ModelInstance</td>
</tr>
<tr>
<td>void SetType (Type t)</td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td>void SetLocalBoundingBox (const Math::bbox &amp;b)</td>
<td>set the local space bounding box</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>virtual Ptr &lt; GraphicsEntity &gt; CreateClone () const</td>
<td>create a clone of the entity</td>
</tr>
<tr>
<td>virtual void OnAttachToStage (const Ptr&lt; Stage &gt; &amp;stage)</td>
<td>called when attached to Stage</td>
</tr>
<tr>
<td>virtual void OnRemoveFromStage ()</td>
<td>called when removed from Stage</td>
</tr>
<tr>
<td>virtual void OnAttachToCell (const Ptr&lt; Cell &gt; &amp;cell)</td>
<td>called when attached to a Cell</td>
</tr>
<tr>
<td>virtual void OnRemoveFromCell ()</td>
<td>called when removed from a Cell</td>
</tr>
<tr>
<td>virtual void OnShow ()</td>
<td>called when the entity becomes visible</td>
</tr>
<tr>
<td>virtual void OnHide ()</td>
<td>called when the entity becomes invisible</td>
</tr>
<tr>
<td>virtual void OnRender ()</td>
<td>called before the entity is rendered</td>
</tr>
<tr>
<td>void UpdatePositionInCellTree ()</td>
<td>update our position in the cell hierarchy</td>
</tr>
<tr>
<td>void UpdateGlobalBoundingBox ()</td>
<td>update the global bounding box from the transform and local box</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Graphics::ActorEntity::SetBaseAnimation(
    const Util::String & anim,
    Timing::Time fadeIn = 0.0,
    Timing::Time timeOffset = 0.0,
    bool onlyIfInactive = true,
    bool waitForOverlayAnim = false,
    float randomValue = 0.0f
)
```

set the current base animation (usually loop anims) - OBSOLETE, use `FadeAnimation()`, `ClearAnimations()`

Set a new base animation. This is usually a looping animation, like Idle, Walking, Running, etc...

**Parameters:**
- `anim` new base animation
- `fadeIn` time to fade from current animation
- `timeOffset` optional animation time offset

```cpp
void Graphics::ActorEntity::SetOverlayAnimation(
    const Util::String & anim,
    Timing::Time fadeIn = 0.0,
    Timing::Time overrideDuration = 0.0,
    bool onlyIfInactive = true,
    float randomValue = 0.0f
)
```

set the current overlay animation (usually oneshot anims) - OBSOLETE, use `FadeAnimation()`, `ClearAnimations()`

Set a new overlay animation. This is usually a looping animation, like Idle, Walking, Running, etc...
Parameters:

- `animName` new base animation
- `fadeIn` time to fade from current animation
- `overrideDuration` if != 0.0, override the animation's duration with this value

```cpp
void Graphics::ActorEntity::SetFadeAnimationMix(const Util::Array<Util::String> &anims, const Util::Array<float> &weights, Timing::Time fadeIn, Timing::Time overrideDuration, float randomValue)
```

set the current fade animations - OBSOLETE, use `FadeAnimation()`, `ClearAnimations()`

Parameters:

- `clipNames` new fade animations
- `weights` new fade animation weights

```cpp
void Graphics::ActorEntity::OnUpdate() [virtual]
```

called before rendering happens

This method checks whether the current overlay animation is over and the base animation must be re-activated. This is the only place where animations are actually started to prevent stuttering when several animations are started in the same frame.

Reimplemented from `Graphics::ModelEntity`.

```cpp
Char::Character * Graphics::ActorEntity::GetCharacterPointer() const [inline]
```

going pointer to our nCharacter2 object

Returns the pointer to our nCharacter2 object.
void Graphics::ActorEntity::EvaluateSkeleton(bool enforceEvaluation = false)

bring the character's skeleton up to date

This brings the character's skeleton up to date. Make sure the entity's time and animation state weights in the rendercontext are up to date before calling this method, to avoid one-frame-latencies.

int Graphics::ActorEntity::GetJointIndexByName(const Util::String &name)

return joint index by name, return InvalidIndex if joint doesn't exist

Returns a character joint index by its name. Returns -1 if a joint by that name doesn't exist in the character.

CharJoint * Graphics::ActorEntity::GetJoint(int jointIndex)

return joint at index

Return pointer to joint or 0 if joint doesn't exist.

const matrix44 & Graphics::ActorEntity::GetLocalJointMatrix(int jointIndex)

return a joint's current matrix in model space

Returns a joint's current matrix in model space. Make sure to call EvaluateSkeleton() before!

matrix44 Graphics::ActorEntity::GetJointMatrixByIndex(IndexT jointIndex)

get joint matrix by index in global space

Find the joint matrix in world space by index. Careful, this method does an EvaluateSkeleton() per invocation!
get joint matrix by name in global space, throws assert if joint doesn't exist

Returns a joint matrix in world space by name. Careful, this method does an \texttt{EvaluateSkeleton()} per invocation!

```c++
void Graphics::ActorEntity::FadeAnimation( const Util::String & anim,
                                          float targetWeight = 0.0f,
                                          Timing::Time fadeInTime = 0.0,
                                          Timing::Time fadeOutTime = 0.0,
                                          float randomValue = 0.0f,
                                          float runTime = 0.0f,
                                          float timeFactor = 1.0f,
                                          Timing::Time sampleTime = 0.0,
                                          bool animRestart = false,
                                          bool fadeOutRunningAnims = true
)
```

fade an animation to a weight

Fades an animation to the specified target weight

**Parameters:**

- \texttt{animRestart} enable this, if you want to restart the clip (if it runs already) before fading it to the target weight
- \texttt{fadeOutRunningAnims} enable this, if all currently running clips should fade out

```c++
void Graphics::ActorEntity::FadeAnimationMix( const Util::Array< Util::String > & anims,
                                              float targetWeight = 0.0f,
                                              Timing::Time fadeInTime = 0.0,
                                              Timing::Time fadeOutTime = 0.0,
                                              float randomValue = 0.0f,
                                              float runTime = 0.0f,
                                              float timeFactor = 1.0f,
                                              Timing::Time sampleTime = 0.0,
)
```
bool animRestart = false,
bool fadeOutRunningAnims = true
)

fade a mix of animations to a weight

Fades a mix of animations to the specified target weight

**Parameters:**
- `animRestart` enable this, if you want to restart currently running clips before fading them to the target weight
- `fadeOutRunningAnims` enable this, if all currently running clips should fade out

```cpp
void Graphics::ActorEntity::FadeAnimationMix(
    const Util::Array<Util::String> &anims,
    const Util::Array<float> &targetWeights,
    Timing::Time fadeInTime = 0.0,
    Timing::Time fadeOutTime = 0.0,
    float randomValue = 0.0f,
    float runTime = 0.0f,
    float timeFactor = 1.0f,
    Timing::Time sampleTime = 0.0,
    bool animRestart = false,
    bool fadeOutRunningAnims = true
)
```

fade a mix of animations to various weights

Fades a mix of animations to the specified target weights

**Parameters:**
- `animRestart` enable this, if you want to restart currently running clips before fading them to the target weights
- `fadeOutRunningAnims` enable this, if all currently running clips should fade out

```cpp
void Graphics::ActorEntity::FadeRunningAnimationsOut()
```
fade all running animations out

Fade out all animation clips with a weight > 0.0f

```cpp
void Graphics::ActorEntity::FadeRunningAnimationsOut ( AnimationType type,
const Util::Array< int > * clipFilter = 0 )
```

fades running clips of the given type out

Fade out all animation clips with a weight > 0.0f, if they are running as Base- or Overlay-Animation

**Parameters:**

- `type` specify which animation type should fade out (Base- or Overlay-Animations)

```cpp
void Graphics::ModelEntity::OnTransformChanged ( ) [protected, virtual, inherited]
```

called when transform matrix changed

In **OnTransformChanged()** we need to update the transformation of our ModelInstance (if it has already been initialised).

Reimplemented from **Graphics::GraphicsEntity**.

```cpp
void Graphics::ModelEntity::OnRenderDebug ( ) [protected, virtual, inherited]
```

called to render a debug visualization of the entity

Render our debug visualization (the bounding box).

Reimplemented from **Graphics::GraphicsEntity**.

```cpp
void Graphics::ModelEntity::ValidateModelInstance ( ) [protected, inherited]
```

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
ClipStatus::Type Graphics::GraphicsEntity::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 Graphics::CameraEntity, Lighting::GlobalLightEntity, and Lighting::SpotLightEntity.

```cpp
Ptr< GraphicsEntity > Graphics::GraphicsEntity::CreateClone() const [protected, virtual, inherited]
```

create a clone of the entity

This method is called to create an exact clone of this graphics entity. The new entity will not be attached to a stage.

```cpp
void Graphics::GraphicsEntity::OnAttachToStage(const Ptr< Stage > & 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 Graphics::GraphicsEntity::OnRemoveFromStage() [protected, virtual, inherited]
```

called when removed from Stage
This method is called when the graphics entity is removed from a stage.

```cpp
void Graphics::GraphicsEntity::OnAttachToCell(const Ptr<Cell> & c) [protected, virtual, inherited]
```
called when attached to a **Cell**

This method is called when the graphics entity is attached to a cell inside a stage. When entity travel through the graphics world, they will be remove and attached themselves from and to Cells as they cross **Cell** borders.

```cpp
void Graphics::GraphicsEntity::OnRemoveFromCell( ) [protected, virtual, inherited]
```
called when removed from a **Cell**

Called when the graphics entity is removed from a cell inside a stage.

```cpp
void Graphics::GraphicsEntity::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.

```cpp
void Graphics::GraphicsEntity::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.

```cpp
void Graphics::GraphicsEntity::OnRender( ) [protected, virtual, inherited]
```
called before the entity is rendered
This method is called on the graphics entity to render itself. This method will only be called if the entity is visible through the camera of the currently rendered View.

`void Graphics::GraphicsEntity::UpdatePositionInCellTree ( ) [protected, inherited]`

update our position in the cell hierarchy

This method is called from `OnUpdate()` when either the entity's transformation or bounding box has changed. It checks if the entity still fits into its current cell, and if not, moves the entity in a new cell.

`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.

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

dump 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

#include <cameraentity.h>

Inheritance diagram for Graphics::CameraEntity:
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>Type</th>
<th>entity types</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>LinkType</td>
<td>visibility link types</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CameraEntity ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~CameraEntity ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>bool <strong>IsAttachedToView () const</strong></td>
<td>return true if camera is attached to a View</td>
</tr>
<tr>
<td>virtual <strong>Math::ClipStatus::Type ComputeClipStatus (const Math::bbox &amp;box)</strong></td>
<td>compute clip status against bounding box</td>
</tr>
<tr>
<td>void <strong>SetupPerspectiveFov</strong> (float fov, float aspect, float zNear, float zFar)</td>
<td>setup camera for perspective field-of-view projection transform</td>
</tr>
<tr>
<td>void <strong>SetupOrthogonal</strong> (float w, float h, float zNear, float zFar)</td>
<td>setup camera for orthogonal projection transform</td>
</tr>
<tr>
<td>const <strong>Math::matrix44 &amp; GetProjTransform () const</strong></td>
<td>get projection matrix</td>
</tr>
<tr>
<td>const <strong>Math::matrix44 &amp; GetViewTransform () const</strong></td>
<td>get view transform (inverse transform)</td>
</tr>
<tr>
<td>const <strong>Math::matrix44 &amp; GetViewProjTransform ()</strong></td>
<td>get view projection matrix (non-const!)</td>
</tr>
<tr>
<td>bool <strong>IsPerspective () const</strong></td>
<td>return true if this is a perspective projection</td>
</tr>
<tr>
<td>bool <strong>IsOrthogonal () const</strong></td>
<td>return true if this is an orthogonal transform</td>
</tr>
<tr>
<td>float <strong>GetZNear () const</strong></td>
<td>get near plane distance</td>
</tr>
<tr>
<td>float <strong>GetZFar () const</strong></td>
<td>get far plane distance</td>
</tr>
<tr>
<td>float <strong>GetFov () const</strong></td>
<td>get field-of-view (only if perspective)</td>
</tr>
<tr>
<td>float <strong>GetAspectRatio () const</strong></td>
<td>get aspect ratio (only if perspective)</td>
</tr>
<tr>
<td>float <strong>GetNearWidth () const</strong></td>
<td></td>
</tr>
</tbody>
</table>
get width of near plane

float **GetNearHeight** () const
get height of near plane

float **GetFarWidth** () const
get width of far plane

float **GetFarHeight** () const
get height of far plane

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 for rendering)

**Type**

**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< Stage > & **GetStage** () const
get the stage this entity is attached to

bool **IsAttachedToStage** () const
return true if entity is attached to stage

const Ptr< Cell > & **GetCell** () const
get the cell this entity is attached to

bool **IsAttachedToCell** () const
return true if entity is attached to cell

const Math::bbox & **GetLocalBoundingBox** () const
get the local space bounding box

const Math::bbox & **GetGlobalBoundingBox** ()
get bounding box in global space

void **ClearLinks** (LinkType linkType)
clear all visibility links

AddLink (LinkType linkType, const Ptr<
<table>
<thead>
<tr>
<th>void</th>
<th><strong>GraphicsEntity</strong> &gt; &amp;entity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>add visibility link</td>
<td></td>
</tr>
<tr>
<td>const <strong>Util::Array</strong> &lt; <strong>Ptr</strong> &lt; <strong>GraphicsEntity</strong> &gt; &gt; &amp;</td>
<td><strong>GetLinks</strong> (LinkType type) const</td>
</tr>
<tr>
<td>get visibility links by type</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><strong>SetTime</strong> (Timing::Time t)</td>
</tr>
<tr>
<td>set graphics time</td>
<td></td>
</tr>
<tr>
<td><strong>Timing::Time</strong></td>
<td><strong>GetTime</strong> () const</td>
</tr>
<tr>
<td>get graphics time</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td><strong>GetRefCount</strong> () const</td>
</tr>
<tr>
<td>get the current refcount</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><strong>AddRef</strong> ()</td>
</tr>
<tr>
<td>increment refcount by one</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><strong>Release</strong> ()</td>
</tr>
<tr>
<td>decrement refcount and destroy object if refcount is zero</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><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>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><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>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><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>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><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>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><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>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><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>
<td></td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></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><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>virtual void <code>OnDeactivate()</code></td>
<td>called before entity is destroyed</td>
</tr>
<tr>
<td>void <code>OnAttachToView</code> (const <code>Ptr&lt; View &gt; &amp;view</code>)</td>
<td>called by <code>View</code> when camera is attached to that view</td>
</tr>
<tr>
<td>void <code>OnRemoveFromView</code> (const <code>Ptr&lt; View &gt; &amp;view</code>)</td>
<td>called by <code>View</code> when camera becomes detached from view</td>
</tr>
<tr>
<td>virtual void <code>OnTransformChanged()</code></td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td>void <code>UpdateViewProjMatrix()</code></td>
<td>update the view projection matrix</td>
</tr>
<tr>
<td>void <code>SetType</code> (Type t)</td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td>void <code>SetLocalBoundingBox</code> (const <code>Math::bbox &amp;b</code>)</td>
<td>set the local space bounding box</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>virtual <code>Ptr &lt; GraphicsEntity &gt;</code> <code>CreateClone</code> () const</td>
<td>create a clone of the entity</td>
</tr>
<tr>
<td>virtual void <code>OnActivate()</code></td>
<td>called when entity is created</td>
</tr>
<tr>
<td>virtual void <code>OnAttachToStage</code> (const <code>Ptr&lt; Stage &gt; &amp;stage</code>)</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>OnAttachToCell</code> (const <code>Ptr&lt; Cell &gt; &amp;cell</code>)</td>
<td>called when attached to a <code>Cell</code></td>
</tr>
<tr>
<td>virtual void <code>OnRemoveFromCell</code> ()</td>
<td>called when removed from a <code>Cell</code></td>
</tr>
<tr>
<td>virtual void <code>OnShow</code> ()</td>
<td>called when the entity becomes visible</td>
</tr>
<tr>
<td>virtual void <code>OnHide</code> ()</td>
<td>called when the entity becomes invisible</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>OnUpdate()</strong></td>
<td>called to update the entity before rendering</td>
</tr>
<tr>
<td>virtual void <strong>OnRender()</strong></td>
<td>called before the entity is rendered</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>UpdatePositionInCellTree()</strong></td>
<td>update our position in the cell hierarchy</td>
</tr>
<tr>
<td>void <strong>UpdateGlobalBoundingBox()</strong></td>
<td>update the global bounding box from the transform and local box</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
ClipStatus::Type Graphics::CameraEntity::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 `Graphics::GraphicsEntity`.

```cpp
void Graphics::CameraEntity::SetupPerspectiveFov(float fov_, float aspect_, float zNear_, float zFar_)
```

setup camera for perspective field-of-view projection transform

Setup camera as perspective projection.

```cpp
void Graphics::CameraEntity::SetupOrthogonal(float w, float h, float zNear_, float zFar_)
```

setup camera for orthogonal projection transform

Setup camera as orthogonal projection.

```cpp
void Graphics::CameraEntity::OnTransformChanged()
```

called when transform matrix changed

We need to keep track of modifications of the transformation matrix.
Reimplemented from **Graphics::GraphicsEntity**.

```cpp
void
Graphics::CameraEntity::UpdateViewProjMatrix( ) [protected]
```

update the view projection matrix

Updates the view-projection matrix.

```cpp
Ptr< GraphicsEntity >
Graphics::GraphicsEntity::CreateClone( ) const [protected, virtual, inherited]
```

create a clone of the entity

This method is called to create an exact clone of this graphics entity. The new entity will not be attached to a stage.

```cpp
void
Graphics::GraphicsEntity::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 **Graphics::ActorEntity**, and **Graphics::ModelEntity**.

```cpp
void
Graphics::GraphicsEntity::OnAttachToStage(const Ptr< Stage > & 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.
called when removed from **Stage**

This method is called when the graphics entity is removed from a stage.

```cpp
void Graphics::GraphicsEntity::OnRemoveFromStage() [protected, virtual, inherited]
```

called when attached to a **Cell**

This method is called when the graphics entity is attached to a cell inside a stage. When entity travel through the graphics world, they will be remove and attached themselves from and to Cells as they cross **Cell** borders.

```cpp
void Graphics::GraphicsEntity::OnAttachToCell(const Ptr<Cell>& c) [protected, virtual, inherited]
```

called when removed from a **Cell**

Called when the graphics entity is removed from a cell inside a stage.

```cpp
void Graphics::GraphicsEntity::OnRemoveFromCell() [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.

```cpp
void Graphics::GraphicsEntity::OnShow() [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.

```cpp
void Graphics::GraphicsEntity::OnHide() [protected, virtual, inherited]
```
void Graphics::GraphicsEntity::OnUpdate() [protected, virtual, inherited]
called to update the entity before rendering

This method is called on the graphics entity to update itself, for instance to implement hierarchy animation or particle updates.

Reimplemented in Graphics::ActorEntity, and Graphics::ModelEntity.

void Graphics::GraphicsEntity::OnRender() [protected, virtual, inherited]
called before the entity is rendered

This method is called on the graphics entity to render itself. This method will only be called if the entity is visible through the camera of the currently rendered View.

void Graphics::GraphicsEntity::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 Graphics::ModelEntity.

void Graphics::GraphicsEntity::UpdatePositionInCellTree() [protected, inherited]
update our position in the cell hierarchy

This method is called from OnUpdate() when either the entity's transformation or bounding box has changed. It checks if the entity still fits into its current cell, and if not, moves the entity in a new cell.

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::Cell
Graphics::Cell Class Reference

#include <cell.h>

Inheritance diagram for Graphics::Cell:
Detailed Description

Hierarchies of Cell objects group graphics entities by spatial relationship. They are the key class for efficient visibility queries. An application may derive specialized subclasses of Cell which must only adhere to the following 2 simple rules:

- if the Cell object is fully visible, all child Cells and all Entities attached to the Cell are guaranteed to be visible
- if the Cell object is fully invisible, all child Cells and all Entities attached to the Cell 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) 2007 Radon Labs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cell ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~Cell ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual void OnAttachToStage (const Ptr&lt; Stage &gt; &amp;stage)</strong></td>
<td>called when attached to <strong>Stage</strong></td>
</tr>
<tr>
<td><strong>virtual void OnRemoveFromStage ()</strong></td>
<td>called when removed from <strong>Stage</strong></td>
</tr>
<tr>
<td><strong>bool IsAttachedToStage () const</strong></td>
<td>return true if currently attached to a stage</td>
</tr>
<tr>
<td><strong>const Ptr&lt; Stage &gt; &amp; GetStage () const</strong></td>
<td>get stage we are attached to</td>
</tr>
<tr>
<td><strong>void SetBoundingBox (const Math::bbox &amp;box)</strong></td>
<td>set the Cell's world space bounding box</td>
</tr>
<tr>
<td><strong>const Math::bbox &amp; GetBoundingBox () const</strong></td>
<td>get the Cell's world space transform</td>
</tr>
<tr>
<td><strong>void AttachChildCell (const Ptr&lt; Cell &gt; &amp;cell)</strong></td>
<td>add a child cell (only during setup phase)</td>
</tr>
<tr>
<td><strong>const Ptr&lt; Cell &gt; &amp; GetParentCell () const</strong></td>
<td>get pointer to parent cell (returns invalid pointer if this is root cell)</td>
</tr>
<tr>
<td><strong>const Util::Array&lt; Ptr&lt; Cell &gt; &gt; &amp; GetChildCells () const</strong></td>
<td>get current child cells</td>
</tr>
<tr>
<td><strong>void AttachEntity (const Ptr&lt; GraphicsEntity &gt; &amp;entity)</strong></td>
<td>attach a graphics entity to this <strong>Cell</strong></td>
</tr>
<tr>
<td><strong>void RemoveEntity (const Ptr&lt; GraphicsEntity &gt; &amp;entity)</strong></td>
<td>remove a graphics entity from this <strong>Cell</strong></td>
</tr>
<tr>
<td><strong>void InsertEntity (const Ptr&lt; GraphicsEntity &gt; &amp;entity)</strong></td>
<td>insert an entity into the <strong>Cell</strong> hierarchy</td>
</tr>
<tr>
<td><strong>const Util::Array&lt; Ptr GraphicsEntity &gt; &amp; GetEntities () const</strong></td>
<td>get current entities</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>get all entities</code></td>
<td>&lt; GraphicsEntity &gt; &amp;</td>
</tr>
<tr>
<td><code>get entities by entity type</code></td>
<td>const Util::Array&lt; Ptr&lt; GraphicsEntity &gt; &amp;&gt; GetEntitiesByType (GraphicsEntity::Type type) const</td>
</tr>
<tr>
<td><code>get the number of entities in hierarchy</code></td>
<td>SizeT GetNumEntitiesInHierarchy () const</td>
</tr>
<tr>
<td><code>get the number of entities in hierarchy by type</code></td>
<td>SizeT GetNumEntitiesInHierarchyByType (GraphicsEntity::Type type) const</td>
</tr>
<tr>
<td><code>get the number of entities in hierarchy by entity type mask</code></td>
<td>SizeT GetNumEntitiesInHierarchyByTypeMask (uint entityTypeMask) const</td>
</tr>
<tr>
<td><code>recursively create visibility links between observers and observed entities</code></td>
<td>void UpdateLinks (const Ptr&lt; GraphicsEntity &gt;&amp; observerEntity, uint entityTypeMask, GraphicsEntity::LinkType linkType)</td>
</tr>
<tr>
<td><code>get the current refcount</code></td>
<td>int GetRefCount () const</td>
</tr>
<tr>
<td><code>increment refcount by one</code></td>
<td>void AddRef ()</td>
</tr>
<tr>
<td><code>decrement refcount and destroy object if refcount is zero</code></td>
<td>void Release ()</td>
</tr>
<tr>
<td><code>return true if this object is instance of given class</code></td>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td><code>return true if this object is instance of given class by string</code></td>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
</tr>
<tr>
<td><code>return true if this object is instance of given class by fourcc</code></td>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td><code>return true if this object is instance of given class, or a derived class</code></td>
<td>bool IsA (const Rtti &amp;rtti) const</td>
</tr>
<tr>
<td><code>return true if this object is instance of given class, or a derived class by string</code></td>
<td>bool IsA (const Util::String &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>------</td>
<td>--------------------------------------------------</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>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>
void Graphics::Cell::AttachChildCell(const Ptr<Cell> &cell)

add a child cell (only during setup phase)

NOTE: the cell hierarchy may only be built during the setup phase while the cell hierarchy haven't been added to the stage yet.

void Graphics::Cell::AttachEntity(const Ptr<GraphicsEntity> &entity)

attach a graphics entity to this Cell

Attach an entity to this Cell. This will happen when a graphics entity moves through the world, leaving and entering cells as necessary.

void Graphics::Cell::InsertEntity(const Ptr<GraphicsEntity> &entity)

insert an entity into the Cell hierarchy

Insert a dynamic graphics entity into the cell tree. The entity will correctly be inserted into the smallest enclosing cell in the tree. The cell may not be currently attached to a cell, the refcount of the entity will be incremented.

Parameters:

  entity pointer to a graphics entity

void Graphics::Cell::UpdateLinks(const Ptr<GraphicsEntity> &observerEntity,
                                 const GraphicsEntity::LinkType linkType
                                 )

  observerEntity, linked type
recursively create visibility links between observers and observed entities

Frontend method for updating visibility links. This method simply calls RecurseUpdateLinks() which recurses into child cells if necessary.

```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::GraphicsEntity
Graphics::GraphicsEntity Class Reference

#include <graphicsentity.h>

Inheritance diagram for Graphics::GraphicsEntity:
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 Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>entity types</td>
</tr>
<tr>
<td>LinkType</td>
<td>visibility link types</td>
</tr>
</tbody>
</table>
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GraphicsEntity()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~GraphicsEntity()</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 currently valid (ready for rendering)</td>
</tr>
<tr>
<td>Type 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;Stage&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>const Ptr&lt;Cell&gt; &amp; GetCell() const</td>
<td>get the cell this entity is attached to</td>
</tr>
<tr>
<td>bool IsAttachedToCell() const</td>
<td>return true if entity is attached to cell</td>
</tr>
<tr>
<td>const Math::bbox &amp; GetLocalBoundingBox() const</td>
<td>get the local space bounding box</td>
</tr>
<tr>
<td>const Math::bbox &amp; GetGlobalBoundingBox()</td>
<td>get bounding box in global space</td>
</tr>
<tr>
<td>void ClearLinks(LinkType linkType)</td>
<td>clear all visibility links</td>
</tr>
</tbody>
</table>
void AddLink (LinkType linkType, const Ptr<GraphicsEntity> &entity)
    add visibility link

const Util::Array<Ptr<GraphicsEntity>> & 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 SetTime (Timing::Time t)
    set graphics time

Timing::Time GetTime () const
    get graphics time

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 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><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>SetType</strong> (<strong>Type</strong> t)</td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td>void <strong>SetLocalBoundingBox</strong> (const <strong>Math::bbox</strong> &amp;b)</td>
<td>set the local space bounding box</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>virtual <strong>Ptr</strong>&lt;<strong>GraphicsEntity</strong>&gt; <strong>CreateClone</strong> () const</td>
<td>create a clone of the entity</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 <strong>Ptr</strong>&lt;Stage&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>OnAttachToCell</strong> (const <strong>Ptr</strong>&lt;Cell&gt; &amp;cell)</td>
<td>called when attached to a Cell</td>
</tr>
<tr>
<td>virtual void <strong>OnRemoveFromCell</strong> ()</td>
<td>called when removed from a Cell</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>OnTransformChanged</strong> ()</td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td>virtual void <strong>OnUpdate</strong> ()</td>
<td>called to update the entity before rendering</td>
</tr>
<tr>
<td>virtual void <strong>OnRender</strong> ()</td>
<td>called before the entity is rendered</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</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>UpdatePositionInCellTree()</code></td>
<td>Update our position in the cell hierarchy</td>
</tr>
<tr>
<td><code>void UpdateGlobalBoundingBox()</code></td>
<td>Update the global bounding box from the transform and local box</td>
</tr>
</tbody>
</table>
Member Function Documentation

**ClipStatus::Type**

```cpp
Graphics::GraphicsEntity::ComputeClipStatus (const Math::bbox& box) [virtual]
```

calculate 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 **Graphics::CameraEntity**, **Lighting::GlobalLightEntity**, and **Lighting::SpotLightEntity**.

**Ptr< GraphicsEntity >**

```cpp
Graphics::GraphicsEntity::CreateClone ( ) const [protected, virtual]
```

created a clone of the entity

This method is called to create an exact clone of this graphics entity. The new entity will not be attached to a stage.

**void**

```cpp
Graphics::GraphicsEntity::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 **Graphics::ActorEntity**, and **Graphics::ModelEntity**.

**void**

```cpp
Graphics::GraphicsEntity::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 `Graphics::ActorEntity`, `Graphics::CameraEntity`, and `Graphics::ModelEntity`.

```cpp
void Graphics::GraphicsEntity::OnAttachToStage(const Ptr<Stage> & 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.

```cpp
void Graphics::GraphicsEntity::OnRemoveFromStage() [protected, virtual]
```
called when removed from `Stage`

This method is called when the graphics entity is removed from a stage.

```cpp
void Graphics::GraphicsEntity::OnAttachToCell(const Ptr<Cell> & c) [protected, virtual]
```
called when attached to a `Cell`

This method is called when the graphics entity is attached to a cell inside a stage. When entity travel through the graphics world, they will be remove and attached themselves from and to Cells as they cross `Cell` borders.

```cpp
void Graphics::GraphicsEntity::OnRemoveFromCell() [protected, virtual]
```
called when removed from a **Cell**

Called when the graphics entity is removed from a cell inside a stage.

```cpp
void Graphics::GraphicsEntity::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.

```cpp
void Graphics::GraphicsEntity::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.

```cpp
void Graphics::GraphicsEntity::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.


```cpp
void Graphics::GraphicsEntity::OnUpdate( ) [protected, virtual]
```

called to update the entity before rendering

This method is called on the graphics entity to update itself, for instance to implement hierarchy animation or particle updates.
Reimplemented in **Graphics::ActorEntity**, and **Graphics::ModelEntity**.

```cpp
void
Graphics::GraphicsEntity::OnRender ( ) [protected, virtual]
```
called before the entity is rendered

This method is called on the graphics entity to render itself. This method will only be called if the entity is visible through the camera of the currently rendered **View**.

```cpp
void
Graphics::GraphicsEntity::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 **Graphics::ModelEntity**.

```cpp
void
Graphics::GraphicsEntity::UpdatePositionInCellTree ( ) [protected]
```
update our position in the cell hierarchy

This method is called from **OnUpdate()** when either the entity's transformation or bounding box has changed. It checks if the entity still fits into its current cell, and if not, moves the entity in a new cell.

```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:

```
Core::RefCounted

Graphics::GraphicsServer
```
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>GraphicsServer()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~GraphicsServer()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <code>Open()</code></td>
<td>Open the graphics server</td>
</tr>
<tr>
<td>void <code>Close()</code></td>
<td>Close the graphics server</td>
</tr>
<tr>
<td>bool <code>IsOpen()</code> const</td>
<td>Return true if graphics server is open</td>
</tr>
<tr>
<td><code>Ptr&lt;Stage&gt;</code> <code>CreateStage</code>(const <code>Util::StringAtom</code> &amp;name, const <code>Ptr&lt;StageBuilder&gt;</code> &amp;stageBuilder)</td>
<td>Create a stage object</td>
</tr>
<tr>
<td>void <code>DiscardStage</code>(const <code>Ptr&lt;Stage&gt;</code> &amp;stage)</td>
<td>Discard a stage object</td>
</tr>
<tr>
<td>void <code>DiscardAllStages()</code></td>
<td>Discard all stage objects</td>
</tr>
<tr>
<td>bool <code>HasStage</code>(const <code>Util::StringAtom</code> &amp;name) const</td>
<td>Return true if a stage exists by name</td>
</tr>
<tr>
<td>const <code>Ptr&lt;Stage&gt;</code> &amp; <code>GetStageByName</code>(const <code>Util::StringAtom</code> &amp;name) const</td>
<td>Lookup a stage by name</td>
</tr>
<tr>
<td>const <code>Util::Array&lt;Ptr&lt;Stage&gt;&gt;</code> &amp; <code>GetStages</code>() const</td>
<td>Get all stages</td>
</tr>
<tr>
<td><code>Ptr&lt;View&gt;</code> <code>CreateView</code>(const <code>Core::Rtti</code> &amp;viewClass, const <code>Util::StringAtom</code> &amp;name, bool isDefaultView=false)</td>
<td>Create a view object</td>
</tr>
<tr>
<td>void <code>DiscardView</code>(const <code>Ptr&lt;View&gt;</code> &amp;view)</td>
<td>Discard a view object</td>
</tr>
<tr>
<td>void <code>DiscardAllViews</code>()</td>
<td>Discard all view objects</td>
</tr>
<tr>
<td>bool <code>HasView</code>(const <code>Util::StringAtom</code> &amp;name) const</td>
<td>Return true if a view exists by name</td>
</tr>
</tbody>
</table>
return true if a view exists by name

const Ptr< View > & GetViewByNamed (const Util::StringAtom &name) const
lookup a view by name

const Util::Array< Ptr< View > > & GetViews () const
get all views

void SetDefaultView (const Ptr< View > &defView)
set the default view

const Ptr< View > & GetDefaultView () const
get the default view

IndexT GetFrameCount () const
get unique frame counter, updated in OnFrame()

void OnFrame (Timing::Time curTime)
call per-frame, this renders the default view

Math::line ComputeWorldMouseRay (const Math::float2 &mousePos, float length, const Math::matrix44 &viewMatrix, const Math::matrix44 &invProjMatrix, float nearPlane)
FIXME: this does not belong here!!! move into dedicated RenderUtil class and make static method!

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
<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsA</strong> (const <strong>Util::FourCC</strong> &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>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
<td><strong>GetClassName</strong> () const</td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></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>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
Graphics::GraphicsServer::ComputeWorldMouseRay(
    const Math::float2 & mousePos,
    float length,
    const Math::matrix44 viewMatrix,
    &
    const Math::matrix44 invProjMatrix,
    &
    float nearPlane
)

FIXME: this does not belong here!!! move into dedicated RenderUtil
class and make static method!

Utility function which computes a ray in world space between the eye
and the current mouse position on the near plane.

```
const **Util::String** & Core::RefCounted::GetClassName

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::ModelEntity
Graphics::ModelEntity Class Reference

#include <modelentity.h>

Inheritance diagram for Graphics::ModelEntity:

```
Core::RefCounted

Graphics::GraphicsEntity

Graphics::ModelEntity

Graphics::ActorEntity
```
Detailed Description

Represents a visible graphics object.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>entity types</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enum</th>
<th>LinkType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>visibility link types</td>
</tr>
</tbody>
</table>
**Public Member Functions**

<table>
<thead>
<tr>
<th>Public Member Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ModelEntity ()</strong></td>
</tr>
<tr>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~ModelEntity ()</td>
</tr>
<tr>
<td>destructor</td>
</tr>
<tr>
<td>void SetResourceId (const Resources::ResourceId &amp;resId)</td>
</tr>
<tr>
<td>set the model's resource id</td>
</tr>
<tr>
<td>const Resources::ResourceId &amp; GetResourceId () const</td>
</tr>
<tr>
<td>get the model's resource id</td>
</tr>
<tr>
<td>Resources::Resource::State GetModelResourceState () const</td>
</tr>
<tr>
<td>get the state of the contained managed model resource</td>
</tr>
<tr>
<td>const Ptr&lt; Models::ModelInstance &gt; &amp; GetModellInstance () const</td>
</tr>
<tr>
<td>get pointer to model instance (only valid if already loaded)</td>
</tr>
<tr>
<td>bool IsActive () const</td>
</tr>
<tr>
<td>return true if entity is currently active (is between OnActivate()/OnDeactivate())</td>
</tr>
<tr>
<td>bool IsValid () const</td>
</tr>
<tr>
<td>return true if entity is current valid (ready for rendering)</td>
</tr>
<tr>
<td>Type GetType () const</td>
</tr>
<tr>
<td>get the entity type</td>
</tr>
<tr>
<td>void SetTransform (const Math::matrix44 &amp;m)</td>
</tr>
<tr>
<td>set the entity's world space transform</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; GetTransform () const</td>
</tr>
<tr>
<td>get the entity's world space transform</td>
</tr>
<tr>
<td>void SetVisible (bool b)</td>
</tr>
<tr>
<td>set the entity's visibility</td>
</tr>
<tr>
<td>bool IsVisible () const</td>
</tr>
<tr>
<td>return true if entity is set to visible</td>
</tr>
<tr>
<td>const Ptr&lt; Stage &gt; &amp; GetStage () const</td>
</tr>
<tr>
<td>get the stage this entity is attached to</td>
</tr>
<tr>
<td>bool IsAttachedToStage () const</td>
</tr>
<tr>
<td>return true if entity is attached to stage</td>
</tr>
<tr>
<td>Function</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>getCell</td>
</tr>
<tr>
<td>isAttachedToCell</td>
</tr>
<tr>
<td>GetLocalBoundingBox</td>
</tr>
<tr>
<td>GetGlobalBoundingBox</td>
</tr>
<tr>
<td>ClearLinks (LinkType linkType)</td>
</tr>
<tr>
<td>AddLink (LinkType linkType, const Ptr/GraphicsEntity &amp;entity)</td>
</tr>
<tr>
<td>GetLinks (LinkType type)</td>
</tr>
<tr>
<td>ComputeClipStatus (const Math::bbox &amp;box)</td>
</tr>
<tr>
<td>SetTime (Timing::Time t)</td>
</tr>
<tr>
<td>GetTime</td>
</tr>
<tr>
<td>GetRefCount</td>
</tr>
<tr>
<td>AddRef</td>
</tr>
<tr>
<td>Release</td>
</tr>
<tr>
<td>IsInstanceOf (const Rtti &amp;rtti)</td>
</tr>
<tr>
<td>IsInstanceOf (const Util::String &amp;className)</td>
</tr>
<tr>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC)</td>
</tr>
<tr>
<td>bool</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>bool</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>bool</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Util::FourCC</strong></td>
</tr>
<tr>
<td></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>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>OnTransformChanged</strong> ()</td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td>virtual void <strong>OnUpdate</strong> ()</td>
<td>called to update the entity before rendering</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>ValidateModelInstance</strong> ()</td>
<td>validate the ModelInstance</td>
</tr>
<tr>
<td>void <strong>SetType</strong> (Type t)</td>
<td>set entity type, call in constructor of derived class!</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>SetValid</strong> (bool b)</td>
<td>set to valid state (when the entity becomes ready for rendering)</td>
</tr>
<tr>
<td>virtual <strong>CreateClone</strong> () const</td>
<td>create a clone of the entity</td>
</tr>
<tr>
<td>virtual void <strong>OnAttachToStage</strong> (const Ptr&lt; Stage &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>OnAttachToCell</strong> (const Ptr&lt; Cell &gt; &amp;cell)</td>
<td>called when attached to a Cell</td>
</tr>
<tr>
<td>virtual void <strong>OnRemoveFromCell</strong> ()</td>
<td>called when removed from a Cell</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>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OnRender ()</td>
<td>called before the entity is rendered</td>
</tr>
<tr>
<td>void UpdatePositionInCellTree ()</td>
<td>update our position in the cell hierarchy</td>
</tr>
<tr>
<td>void UpdateGlobalBoundingBox ()</td>
<td>update the global bounding box from the transform and local box</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Graphics::ModelEntity::OnActivate() [protected, virtual]
called when entity is created
This will initialize the managed model inside the ModelEntity.
Reimplemented from Graphics::GraphicsEntity.
Reimplemented in Graphics::ActorEntity.

void Graphics::ModelEntity::OnDeactivate() [protected, virtual]
called before entity is destroyed
Cleanup our managed model, and ModelInstance if it is already initialized.
Reimplemented from Graphics::GraphicsEntity.
Reimplemented in Graphics::ActorEntity.

void Graphics::ModelEntity::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 Graphics::GraphicsEntity.

void Graphics::ModelEntity::OnUpdate() [protected, virtual]
called to update the entity before rendering
In `OnUpdate()` the `ModelEntity` first needs to check whether the ManagedModel has already been loaded, and if yes, a ModelInstance will be created, finally, the ModelInstance will be prepared for rendering.

Reimplemented from `Graphics::GraphicsEntity`.

Reimplemented in `Graphics::ActorEntity`.

```cpp
void Graphics::ModelEntity::OnRenderDebug( ) [protected, virtual]
```
called to render a debug visualization of the entity

Render our debug visualization (the bounding box).

Reimplemented from `Graphics::GraphicsEntity`.

```cpp
void Graphics::ModelEntity::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
ClipStatus::Type Graphics::GraphicsEntity::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 `Graphics::CameraEntity`, `Lighting::GlobalLightEntity`, and `Lighting::SpotLightEntity`.

```cpp
Ptr< GraphicsEntity > Graphics::GraphicsEntity::CreateClone( ) const [protected, virtual, inherited]
```
create a clone of the entity

This method is called to create an exact clone of this graphics entity. The new entity will not be attached to a stage.

```cpp
void Graphics::GraphicsEntity::OnAttachToStage(const Ptr<Stage> & 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 Graphics::GraphicsEntity::OnRemoveFromStage() [protected, virtual, inherited]
```
called when removed from **Stage**

This method is called when the graphics entity is removed from a stage.

```cpp
void Graphics::GraphicsEntity::OnAttachToCell(const Ptr<Cell> & c) [protected, virtual, inherited]
```
called when attached to a **Cell**

This method is called when the graphics entity is attached to a cell inside a stage. When entity travel through the graphics world, they will be remove and attached themselves from and to Cells as they cross **Cell** borders.

```cpp
void Graphics::GraphicsEntity::OnRemoveFromCell() [protected, virtual, inherited]
```
called when removed from a **Cell**
Called when the graphics entity is removed from a cell inside a stage.

```cpp
void Graphics::GraphicsEntity::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.

```cpp
void Graphics::GraphicsEntity::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.

```cpp
void Graphics::GraphicsEntity::OnRender() [protected, virtual, inherited]
```
called before the entity is rendered

This method is called on the graphics entity to render itself. This method will only be called if the entity is visible through the camera of the currently rendered `View`.

```cpp
void Graphics::GraphicsEntity::UpdatePositionInCellTree() [protected, inherited]
```
update our position in the cell hierarchy

This method is called from `OnUpdate()` when either the entity's transformation or bounding box has changed. It checks if the entity still fits into its current cell, and if not, moves the entity in a new cell.

```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.
Graphics::QuadtreeStageBuilder
#include <quadtreestagebuilder.h>

Inheritance diagram for Graphics::QuadtreeStageBuilder:

```
Core::RefCounted

Graphics::StageBuilder

Graphics::QuadtreeStageBuilder
```
Detailed Description

A quadtree stage builder, which sort cells into a quadtree for culling.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>QuadtreeStageBuilder()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>virtual ~QuadtreeStageBuilder()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual void BuildStage(const Ptr&lt;Stage&gt;&amp; stage)</code></td>
<td>Setup the stage</td>
</tr>
<tr>
<td><code>void SetQuadTreeSettings(uchar depth, const Math::bbox&amp; worldBBox)</code></td>
<td>Set quad tree depth and bounding box</td>
</tr>
<tr>
<td><code>void SetAttributes(const Attr::AttributeContainer&amp; attrs)</code></td>
<td>Set stage builder attributes</td>
</tr>
<tr>
<td><code>const Attr::AttributeContainer&amp; GetAttributes() const</code></td>
<td>Get stage builder attributes</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><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</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><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

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

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]

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.
Graphics::<strong>SimpleStageBuilder</strong>
Graphics::SimpleStageBuilder Class Reference

#include <simplestagebuilder.h>

Inheritance diagram for Graphics::SimpleStageBuilder:

```
Core::RefCounted

Graphics::StageBuilder

Graphics::SimpleStageBuilder
```
Detailed Description

The most simple stage builder just creates a single cell where all entities live in. Use this stage builder when you only have one or very few graphics entities where no culling is needed.

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

<table>
<thead>
<tr>
<th>Member Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SimpleStageBuilder ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~SimpleStageBuilder ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>virtual void BuildStage (const Ptr&lt; Stage &gt; &amp;stage)</strong></td>
<td>Setup the stage</td>
</tr>
<tr>
<td><strong>void SetAttributes (const Attr::AttributeContainer &amp;attrs)</strong></td>
<td>Set stage builder attributes</td>
</tr>
<tr>
<td><strong>const Attr::AttributeContainer &amp; GetAttributes () const</strong></td>
<td>Get stage builder attributes</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</code> &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>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>Util::FourCC</code></th>
<th><strong>GetClassFourCC</strong> () 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>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.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

**Graphics::Stage**
#include <stage.h>

Inheritance diagram for Graphics::Stage:
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
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Stage()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~Stage()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>IsAttachedToServer()</code></td>
<td><code>bool</code> return <code>true</code> if currently attached to graphics server</td>
</tr>
<tr>
<td><code>GetName()</code></td>
<td><code>const</code> get human readable name of the stage</td>
</tr>
<tr>
<td><code>GetStageBuilder()</code></td>
<td><code>const</code> get stage builder object</td>
</tr>
<tr>
<td><code>SetRootCell</code></td>
<td><code>void</code> set the root cell of the stage</td>
</tr>
<tr>
<td><code>GetRootCell()</code></td>
<td><code>const</code> get the root cell of the stage</td>
</tr>
<tr>
<td><code>AttachEntity</code></td>
<td><code>virtual</code> void attach an entity to the stage</td>
</tr>
<tr>
<td><code>RemoveEntity</code></td>
<td><code>virtual</code> void remove an entity from the stage</td>
</tr>
<tr>
<td><code>GetEntities()</code></td>
<td><code>const</code> get an array of all entities attached to the stage</td>
</tr>
<tr>
<td><code>GetEntitiesByType</code></td>
<td><code>const</code> get entities by type</td>
</tr>
<tr>
<td><code>UpdateEntities</code></td>
<td><code>virtual</code> void update (animate) entities in the cell</td>
</tr>
<tr>
<td><code>UpdateCameraLinks</code></td>
<td><code>virtual</code> void update camera links for a given camera</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>virtual void <strong>UpdateLightLinks</strong> ()</td>
<td>update light links for all visible lights (after updating camera links!)</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>
<tr>
<td>static void</td>
<td><strong>DumpRefCountingLeaks</strong> ()</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
</tbody>
</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><code>SetName</code> (const <code>Util::StringAtom</code> &amp;name)</td>
<td>set a human readable name on the stage</td>
</tr>
<tr>
<td>void</td>
<td><code>SetStageBuilder</code> (const <code>Ptr&lt; StageBuilder &gt;</code> &amp;stageBuilder)</td>
<td>set stage builder object</td>
</tr>
<tr>
<td>virtual</td>
<td><code>OnAttachToServer</code> ()</td>
<td>called when the stage is attached to graphics server</td>
</tr>
<tr>
<td>virtual</td>
<td><code>OnRemoveFromServer</code> ()</td>
<td>called when the stage is detached from graphics server</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Graphics::Stage::SetRootCell (const Ptr<Cell>& cell)

set the root cell of the stage

Setting a root cell will also initialize it.

void Graphics::Stage::AttachEntity (const Ptr<GraphicsEntity>& 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).

void Graphics::Stage::RemoveEntity (const Ptr<GraphicsEntity>& 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.

void Graphics::Stage::UpdateEntities (Timing::Time curTime, IndexT curFrameCount) [virtual]

update (animate) entities in the cell

Update the entities in the stage. This usually implements graphics
animations, etc... This method must be called first after BeginUpdate() and before any of the visibility link update methods to make sure entity transforms and bounding boxes are updated before resolving visibility.

FIXME: this is no good for worlds with many static entities, entities should know whether they need to have their Update method called, and register themselves somewhere!

```cpp
void Graphics::Stage::UpdateCameraLinks(const Ptr<CameraEntity> 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 Graphics::Stage::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.

```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::StageBuilder
Graphics::StageBuilder Class Reference

#include <stagebuilder.h>

Inheritance diagram for Graphics::StageBuilder:
Detailed Description

**Stage** builders are application-derived classes which construct a stage (build a hierarchy of Cells and populate them with graphics entities).

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StageBuilder ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~StageBuilder ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>SetAttributes</strong> (const Attr::AttributeContainer &amp;attrs)</td>
<td>set stage builder attributes</td>
</tr>
<tr>
<td>const Attr::AttributeContainer &amp; <strong>GetAttributes ()</strong> const</td>
<td>get stage builder attributes</td>
</tr>
<tr>
<td>virtual void <strong>BuildStage</strong> (const Ptr&lt; Stage &gt; &amp;stage)</td>
<td>setup the stage</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>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</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
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::View**
Graphics::View Class Reference

#include <view.h>

Inheritance diagram for Graphics::View:

```
Core::RefCounted
    |
    V
Graphics::View
```
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><strong>View ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~View ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>bool IsAttachedToServer () const</strong></td>
<td>return true if currently attached to graphics server</td>
</tr>
<tr>
<td><strong>const Util::StringAtom &amp; GetName () const</strong></td>
<td>get human-readable name</td>
</tr>
<tr>
<td><strong>void SetStage (const Ptr&lt; Stage &gt; &amp;stage)</strong></td>
<td>set the stage this View is associated with</td>
</tr>
<tr>
<td><strong>const Ptr&lt; Stage &gt; &amp; GetStage () const</strong></td>
<td>get the stage this View is associated with</td>
</tr>
<tr>
<td><strong>void SetCameraEntity (const Ptr&lt; CameraEntity &gt; &amp;camera)</strong></td>
<td>set the CameraEntity this View looks through</td>
</tr>
<tr>
<td><strong>const Ptr&lt; CameraEntity &gt; &amp; GetCameraEntity () const</strong></td>
<td>get the CameraEntity this View looks through</td>
</tr>
<tr>
<td><strong>void SetRenderTarget (const CoreGraphics::RenderTarget &amp;renderTarget)</strong></td>
<td>set the render target this view renders to</td>
</tr>
<tr>
<td><strong>const Ptr&lt; CoreGraphics::RenderTarget &gt; &amp; GetRenderTarget () const</strong></td>
<td>get the render target this view renders to</td>
</tr>
<tr>
<td><strong>void setFrameShader (const Frame::FrameShader &amp;frameShader)</strong></td>
<td>set the view's frame shader</td>
</tr>
<tr>
<td><strong>const Ptr&lt; Frame::FrameShader &gt; &amp; GetFrameShader () const</strong></td>
<td>get the view's frame shader</td>
</tr>
<tr>
<td><strong>void AddDependency (const Ptr&lt; View &gt; &amp;view)</strong></td>
<td>add a view which this view depends on</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetDependencies () const</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>Render ()</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>RenderDebugSimple ()</code></td>
<td>render a debug view of the world, without begin and end shape rendering</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)</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>
</tbody>
</table>
const `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>
</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 SetName (const Util::StringAtom &amp;name)</code></td>
<td>set a human-readable name of the view</td>
</tr>
<tr>
<td><code>virtual void OnAttachToServer ()</code></td>
<td>called when attached to graphics server</td>
</tr>
<tr>
<td><code>virtual void OnRemoveFromServer ()</code></td>
<td>called when detached from graphics server</td>
</tr>
<tr>
<td><code>void ResolveVisibleLights ()</code></td>
<td>resolve visible lights</td>
</tr>
<tr>
<td><code>void ResolveVisibleModelNodeInstances ()</code></td>
<td>resolve visibility for optimal batch rendering</td>
</tr>
</tbody>
</table>
Member Function Documentation

void
Graphics::View::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 Render().

void
Graphics::View::Render ( ) [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 UpdateVisibilityLinks()

void
Graphics::View::RenderDebug ( ) [virtual]

render a debug view of the world

Renders a debug visualization. Can be called alone or after Render().

void
Graphics::View::RenderDebugSimple ( ) [virtual]

render a debug view of the world, without begin and end shape renderering

Renders a debug visualization. Must be called inside Begin and End of shaperenderer.

void
Graphics::View::ResolveVisibleLights ( ) [protected]

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...

void
Graphics::View::ResolveVisibleModelNodeInstances ( ) [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.

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

GraphicsFeature::CameraDistance
GraphicsFeature::CameraDistance
Class Reference

#include <cameradistance.h>

Inheritance diagram for GraphicsFeature::CameraDistance:
Detailed Description

Change the distance of a 3rd camera to its lookat point.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CameraDistance ()</strong></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td><strong>void SetRelativeDistanceChange (float d)</strong></td>
<td>set relative camera distance change (-1.0f..+1.0f)</td>
</tr>
<tr>
<td><strong>float GetRelativeDistanceChange () const</strong></td>
<td>get relative camera distance change</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>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></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) 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
builds only!

This method should be called as the very last before an application exits.
GraphicsFeature::CameraFocus
#include <camerafocus.h>

Inheritance diagram for GraphicsFeature::CameraFocus:
### Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CameraFocus</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>SetObtainFocus</strong> (bool b)</td>
<td>set the direction vector</td>
</tr>
<tr>
<td><strong>GetObtainFocus</strong> () const</td>
<td>get the direction vector</td>
</tr>
<tr>
<td><strong>CheckId</strong> (const Messaging::Id &amp;id) const</td>
<td>return true if message is of the given id</td>
</tr>
<tr>
<td><strong>Encode</strong> (const Ptr&lt; IO::BinaryWriter &gt; &amp;writer)</td>
<td>encode message into a stream</td>
</tr>
<tr>
<td><strong>Decode</strong> (const Ptr&lt; IO::BinaryReader &gt; &amp;reader)</td>
<td>decode message from a stream</td>
</tr>
<tr>
<td><strong>SetHandled</strong> (bool b)</td>
<td>set the handled flag</td>
</tr>
<tr>
<td><strong>Handled</strong> () const</td>
<td>return true if the message has been handled</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)</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)</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</td>
</tr>
</tbody>
</table>
```plaintext
| return true if this object is instance of given class, or a derived class, by string |
| bool | IsA (const Util::FourCC &rttiFourCC) const |
| const Util::String & | GetClassName () const |
| get the class name |
| const 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
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.
GraphicsFeature::CameraOrbit
GraphicsFeature::CameraOrbit Class Reference

#include <cameraorbit.h>

Inheritance diagram for GraphicsFeature::CameraOrbit:
Detailed Description

A camera orbit rotation message.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CameraOrbit ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>SetHorizontalRotation (float a)</strong></td>
<td>set horizontal rotation angle in degree</td>
</tr>
<tr>
<td><strong>GetHorizontalRotation () const</strong></td>
<td>get horizontal rotation angle in degree</td>
</tr>
<tr>
<td><strong>SetVerticalRotation (float a)</strong></td>
<td>set vertical rotation angle in degree</td>
</tr>
<tr>
<td><strong>GetVerticalRotation () const</strong></td>
<td>get vertical rotation angle in degree</td>
</tr>
<tr>
<td><strong>CheckId (const Messaging::Id &amp;id) const</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>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) 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</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

**GraphicsFeature::CameraReset**
GraphicsFeature::CameraReset Class Reference

#include <camerareset.h>

Inheritance diagram for GraphicsFeature::CameraReset:
Detailed Description

A camera reset message.

(C) 2007 Radon Labs GmbH
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool CheckId</code> (const <code>Messaging::Id</code> &amp;id) const</td>
<td>return true if message is of the given id</td>
</tr>
<tr>
<td><code>virtual void Encode</code> (const <code>Ptr&lt; IO::BinaryWriter&gt;</code> &amp;writer)</td>
<td>encode message into a stream</td>
</tr>
<tr>
<td><code>virtual void Decode</code> (const <code>Ptr&lt; IO::BinaryReader&gt;</code> &amp;reader)</td>
<td>decode message from a stream</td>
</tr>
<tr>
<td><code>void SetHandled</code> (bool b)</td>
<td>set the handled flag</td>
</tr>
<tr>
<td><code>bool Handled</code> () const</td>
<td>return true if the message has been handled</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<code> &amp; GetClassName</code> () const</td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () 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><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.
GraphicsFeature::GetGraphicsEntities
GraphicsFeature::GetGraphicsEntities
Class Reference

#include <getgraphicsentities.h>

Inheritance diagram for GraphicsFeature::GetGraphicsEntities:

```
Core::RefCounted

Messaging::Message

GraphicsFeature::GetGraphicsEntities
```
Detailed Description

Returns the graphics entities used by a game entity for rendering.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void SetEntities (const Util::Array&lt; Ptr&lt;Graphics::ModelEntity &gt;&gt; &amp;e)</td>
<td>set graphics entities array</td>
</tr>
<tr>
<td>void SetEntity (const Ptr<a href="">Graphics::ModelEntity</a> &amp;e)</td>
<td>set a single graphics entity</td>
</tr>
<tr>
<td>const Util::Array&lt; Ptr&lt;Graphics::ModelEntity &gt;&gt; &amp; GetEntities() const</td>
<td>get graphics entities array</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<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>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>
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

```cpp
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.
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.

The GraphicsFeatureUnit also offers visualization of debug shapes. Just attach your feature which should be considered in debug rendering. It also allows switching thru all debug renderings of each feature.

Additonal to the rendering the graphicsfeature creates and triggers the input server.

(C) 2007 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

GraphicsFeature::InputFocus
GraphicsFeature::InputFocus Class Reference

#include <inputfocus.h>

Inheritance diagram for GraphicsFeature::InputFocus:
Detailed Description

**Commands** an entity to turn into the specified direction defined by a 3d vector. The direction vector can be absolute or camera relative.

(C) 2007 Radon Labs GmbH
Public Member Functions

**InputFocus** ()

constructor

void **SetObtainFocus** (bool b)

set the direction vector

bool **GetObtainFocus** () const

get the direction vector

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

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
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 string</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>
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.
GraphicsFeature::SetVisible
GraphicsFeature::SetVisible Class Reference

#include <setvisible.h>
Detailed Description

Shows or hides all graphics entities of a game entity.

(C) 2007 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::Base64Writer
Http::Base64Writer Class Reference

#include <base64writer.h>

Inheritance diagram for Http::Base64Writer:

```
Core::RefCounted
    ↓
IO::StreamWriter
    ↓
Http::Base64Writer
```
Detailed Description

StreamWriter which implements a Base64 encoder. Encodes blocks of 3 bytes with binary data into blocks of 4 bytes of Base64 data.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base64Writer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~Base64Writer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td>void *<em>Write (void <em>data, SizeT numBytes)</em></em></td>
<td>Write binary data to stream and apply Base64 encoding (all data must be written at once)</td>
</tr>
<tr>
<td>void <strong>SetStream (const Ptr&lt; Stream &gt; &amp;s)</strong></td>
<td>Set stream to write to</td>
</tr>
<tr>
<td>const Ptr&lt; Stream &gt; &amp; <strong>GetStream () const</strong></td>
<td>Get currently set stream</td>
</tr>
<tr>
<td>bool <strong>HasStream () const</strong></td>
<td>Return true if a stream is set</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 () const</strong></td>
<td>Return true if currently open</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 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 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>dump refcounting leaks, call at end of application (NEBUHALA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Http::Base64Writer::Write( void * data,
                                   SizeT numBytes
                               )

write binary data to stream and apply Base64 encoding (all data must be written at once)

Write a block of data into a base64 encoded stream. NOTE: all data must be written at once!

See: http://base64.sourceforge.net/b64.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.

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.

```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::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><strong>virtual void</strong> HandleRequest (const Ptr&lt; 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><strong>const Util::String</strong> &amp; GetName () const</td>
<td>get a human readable name of the request handler</td>
</tr>
<tr>
<td><strong>const Util::String</strong> &amp; GetDesc () const</td>
<td>get a human readable description of the request handler</td>
</tr>
<tr>
<td><strong>const Util::String</strong> &amp; GetRootLocation () const</td>
<td>get a resource location path which is accepted by the handler (e.g. &quot;/index.html&quot;)</td>
</tr>
<tr>
<td><strong>virtual bool</strong> AcceptsRequest (const Ptr&lt; HttpRequest &gt; &amp;request)</td>
<td>return true if the http request is accepted by the request handler</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 Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td><strong>bool</strong> 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><strong>bool</strong> 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><strong>bool</strong> 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><strong>bool</strong> 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><strong>bool</strong> 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>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>--------------------------------------------------</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>

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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>SetName</code> (const <code>Util::String</code> &amp;n)</td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td>void <code>SetDesc</code> (const <code>Util::String</code> &amp;d)</td>
<td>set human readable description</td>
</tr>
<tr>
<td>void <code>SetRootLocation</code> (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
bool Http::HttpRequestHandler::AcceptsRequest(const Ptr<HttpRequest>& request) [virtual, inherited]

return true if the http request is accepted by the request handler

Overwrite this method in your subclass and decide if you want to react on the request or not.

Reimplemented in Debug::CorePageHandler, Debug::IoPageHandler, Debug::MemoryPageHandler, Debug::ScriptingPageHandler, Debug::DisplayPageHandler, Debug::MeshPageHandler, Debug::ShaderPageHandler, and Debug::TexturePageHandler.
```

```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

Http::HtmlElement
Http::HtmlElement Class Reference

#include <htmllelement.h>
Detailed Description

HTML markup elements.

(C) 2007 Radon Labs GmbH
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>elements</td>
</tr>
</tbody>
</table>
Static Public Member Functions

\texttt{static Util::String ToHtml (Code c)}
\textit{convert to string}

The Nebula Device 3 documentation generated by \texttt{doxygen} at Tue Feb 19 12:16:47 2008
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>HtmlPageWriter ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~HtmlPageWriter ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void setTitle (const Util::String &amp;title)</td>
<td>set the title of the page</td>
</tr>
<tr>
<td>virtual bool Open ()</td>
<td>begin writing the stream</td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td>end writing the stream</td>
</tr>
<tr>
<td>void AddAttr (const Util::String &amp;name, const Util::String &amp;value)</td>
<td>add an attribute for the next element</td>
</tr>
<tr>
<td>void Begin (HtmlElement::Code e)</td>
<td>begin a generic element</td>
</tr>
<tr>
<td>void End (HtmlElement::Code e)</td>
<td>end a generic element</td>
</tr>
<tr>
<td>void Element (HtmlElement::Code e, const Util::String &amp;text)</td>
<td>shortcut for Begin() / Text() / End()</td>
</tr>
<tr>
<td>void LineBreak ()</td>
<td>write a line break</td>
</tr>
<tr>
<td>void HorizontalRule ()</td>
<td>write a horizontal rule</td>
</tr>
<tr>
<td>void Text (const Util::String &amp;t)</td>
<td>add inline text</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; 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 isOpen () const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetRefCount</strong> (int) 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>
**Member Function Documentation**

```cpp
void Http::HtmlPageWriter::AddAttr(
    const Util::String &name,
    const Util::String &value
)
```

add an attribute for the next element

**Add 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.

```cpp
void Http::HtmlPageWriter::Begin(HtmlElement::Code element)
```

begin a generic element

**Begin a generic HTML element.** Elements can be nested.

```cpp
void Http::HtmlPageWriter::End(HtmlElement::Code element)
```

dend a generic element

**End a generic HTML element.** The element must match the last `Begin()` call!

```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 [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.
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
### Public Types

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

<table>
<thead>
<tr>
<th>static</th>
<th>Code</th>
<th>FromString (const Util::String &amp;str)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>convert from string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static</th>
<th>Util::String</th>
<th>ToString (Code c)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>convert to string</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Http::HttpRequest
Http::HttpRequest Class Reference

#include <httprequest.h>

Inheritance diagram for Http::HttpRequest:
Detailed Description

Encapsulates a complete **Http** request into a message.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HttpRequest()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>virtual ~HttpRequest()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>void SetMethod(HttpMethod::Code requestMethod)</code></td>
<td>Set the <em>Http</em> method (<em>GET</em>, <em>PUT</em>, etc...)</td>
</tr>
<tr>
<td><code>HttpMethod::Code GetMethod()</code> const</td>
<td>Get the <em>Http</em> method</td>
</tr>
<tr>
<td><code>void SetURI(const IO::URI &amp;requestUri)</code></td>
<td>Set the request <em>URI</em></td>
</tr>
<tr>
<td><code>const IO::URI &amp; GetURI()</code> const</td>
<td>Get the request <em>URI</em></td>
</tr>
<tr>
<td><code>void SetResponseContentStream(const Ptr&lt;IO::Stream&gt; &amp;responseContentStream)</code></td>
<td>Set the response content stream</td>
</tr>
<tr>
<td><code>const Ptr&lt;IO::Stream&gt; &amp; GetResponseContentStream()</code> const</td>
<td>Get the response content stream</td>
</tr>
<tr>
<td><code>HttpStatus::Code GetStatus()</code> const</td>
<td>Get the http status (set by <em>HttpRequestHandler</em>)</td>
</tr>
<tr>
<td><code>bool CheckId(const Messaging::Id &amp;id)</code> const</td>
<td>Return true if message is of the given <em>id</em></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()</code> const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>return true if the message has been handled</td>
<td>int GetRefCount () 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>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><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.
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:
Detailed Description

HttpRequestHandlers are attached to the HttpRequestHandlers and process incoming HTTP requests. When an HttpRequestHandler receives an Http request, 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>virtual <code>~HttpRequestHandler ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>const <code>Util::String &amp; GetName () const</code></td>
<td>Get a human readable name of the request handler</td>
</tr>
<tr>
<td>const <code>Util::String &amp; GetDesc () const</code></td>
<td>Get a human readable description of the request handler</td>
</tr>
<tr>
<td>const <code>Util::String &amp; GetRootLocation () const</code></td>
<td>Get a resource location path which is accepted by the handler (e.g. /index.html)</td>
</tr>
<tr>
<td>virtual bool <code>AcceptsRequest (const Ptr&lt; HttpRequest &gt; &amp;request)</code></td>
<td>Return true if the http request is accepted by the request handler</td>
</tr>
<tr>
<td>virtual void <code>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>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)</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 (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>Function</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>
</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>SetName</code> (const <code>Util::String &amp;n</code>)</td>
<td>set human readable name of the request handler</td>
</tr>
<tr>
<td><code>SetDesc</code> (const <code>Util::String &amp;d</code>)</td>
<td>set human readable description</td>
</tr>
<tr>
<td><code>SetRootLocation</code> (const <code>Util::String &amp;l</code>)</td>
<td>set the root location of the request handler</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
bool Http::HttpRequestHandler::AcceptsRequest (const Ptr<HttpRequest> & request ) [virtual]
```

return true if the http request is accepted by the request handler

Overwrite this method in your subclass and decide if you want to react on the request or not.

Reimplemented in `Debug::CorePageHandler`, `Debug::IoPageHandler`, `Debug::MemoryPageHandler`, `Debug::ScriptingPageHandler`, `Debug::DisplayPageHandler`, `Debug::MeshPageHandler`, `Debug::ShaderPageHandler`, and `Debug::TexturePageHandler`.

```cpp
void Http::HttpRequestHandler::HandleRequest (const Ptr<HttpRequest> & request ) [virtual]
```

handle a http request, the handler is expected to fill the content stream with response data

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`, `Http::DefaultHttpRequestHandler`, `Debug::IoPageHandler`, `Debug::MemoryPageHandler`, `Debug::ScriptingPageHandler`, `Debug::DisplayPageHandler`, `Debug::MeshPageHandler`, `Debug::ShaderPageHandler`, and `Debug::TexturePageHandler`.

```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:

```
Core::RefCounted

IO::StreamReader

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>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HttpRequestReader</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>bool <strong>ReadRequest</strong> ()</td>
<td>decode the request from the stream (call first before Get methods!)</td>
</tr>
<tr>
<td>bool <strong>IsValidHttpRequest</strong> () const</td>
<td>return true if the stream contains a valid HTTP request</td>
</tr>
<tr>
<td>HttpMethod::Code <strong>GetHttpMethod</strong> () const</td>
<td>get HTTP request method</td>
</tr>
<tr>
<td>const IO::URI &amp; <strong>GetRequestURI</strong> () const</td>
<td>get request URI</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 Ptr&lt; Stream &gt; &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 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
**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

```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]
```

going 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::HttpResponseWriter
Http::HttpResponseWriter Class Reference

#include <httpresponsewriter.h>

Inheritance diagram for Http::HttpResponseWriter:

```
  Core::RefCounted
    |
    v
  IO::StreamWriter
    |
    v
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><code>void SetStatusCode (HttpStatus::Code statusCode)</code></td>
<td>set status code</td>
</tr>
<tr>
<td><code>void SetContent (const Ptr&lt;IO::Stream&gt; &amp;contentStream)</code></td>
<td>set optional content stream (needs valid media type!)</td>
</tr>
<tr>
<td><code>void WriteResponse ()</code></td>
<td>write http response 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 () 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>virtual <code>bool Open ()</code></td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td>virtual <code>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><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>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</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::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.

```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>virtual ~HttpServer ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void SetPort (ushort p)</td>
<td>set port number for http service</td>
</tr>
<tr>
<td>ushort GetPort () const</td>
<td>get port number of http service</td>
</tr>
<tr>
<td>bool Open ()</td>
<td>open the http server</td>
</tr>
<tr>
<td>void Close ()</td>
<td>close the http server</td>
</tr>
<tr>
<td>bool IsOpen () const</td>
<td>return true if server is open</td>
</tr>
<tr>
<td>void AttachRequestHandler (Ptr&lt;HttpRequestHandler&gt; &amp;h)</td>
<td>attach a request handler to the server</td>
</tr>
<tr>
<td>void RemoveRequestHandler (Ptr&lt;HttpRequestHandler&gt; &amp;h)</td>
<td>remove a request handler from the server</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>
</tbody>
</table>
### IsA (const Rtti &rtti) const

Return true if this object is instance of given class, or a derived class.

```cpp
bool IsA (const Rtti &rtti) const
```

### 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::String &rttiName) const
```

### IsA (const Util::FourCC &rttiFourCC) const

Return true if this object is instance of given class, or a derived class, by fourcc.

```cpp
bool IsA (const Util::FourCC &rttiFourCC) const
```

### Util::String GetClassName () const

Get the class name.

```cpp
const Util::String & GetClassName () const
```

### Util::FourCC GetClassFourCC () const

Get the class FourCC code.

```cpp
Util::FourCC GetClassFourCC () const
```
## 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::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

```
enum Code
  status codes
```
**Static Public Member Functions**

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Code</td>
<td><strong>FromString</strong> (const Util::String &amp;str)</td>
<td>convert from string</td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>ToString</strong> (Code c)</td>
<td>convert to string</td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>ToHumanReadableString</strong> (Code c)</td>
<td>convert code to human readable string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:47 2008
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>enum</th>
<th>Button</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>gamepad buttons</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>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>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>return true if this input handler captures input</td>
</tr>
<tr>
<td>int</td>
<td><strong>GetRefCount</strong> () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td><strong>AddRef</strong> ()</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void</td>
<td><strong>Release</strong> ()</td>
<td></td>
</tr>
</tbody>
</table>
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 Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 <strong>OnBeginFrame</strong> ()</td>
<td>called on <code>InputServer::BeginFrame()</code></td>
</tr>
<tr>
<td>void <strong>UpdateButtonState</strong> (const XINPUT_GAMEPAD &amp;curState, WORD xiBtn, <strong>Button</strong> btn)</td>
<td>update the state of a game pad button</td>
</tr>
<tr>
<td>void <strong>UpdateTriggerAxis</strong> (const XINPUT_GAMEPAD &amp;curState, <strong>Axis</strong> axis)</td>
<td>update the state of a trigger axis</td>
</tr>
<tr>
<td>void <strong>UpdateThumbAxis</strong> (const XINPUT_GAMEPAD &amp;curState, <strong>Axis</strong> axis)</td>
<td>update the state of a thumb stick axis</td>
</tr>
<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>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 <code>InputServer::EndFrame()</code></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 bool <strong>OnEvent</strong> (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()

called on InputServer::BeginFrame()

This compares the current state of the game pad against the previous state and sets the internal state accordingly.

Reimplemented from Input::InputHandler.

void XInput::XInputGamePad::UpdateButtonState(const XINPUT_GAMEPAD& curState, WORD xiBtn, Button btn)

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.

void Input::InputHandler::BeginCapture()

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.

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`.

```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.
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.
Input::InputEvent
Input::InputEvent Class Reference

#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 Type</th>
<th>input event types</th>
</tr>
</thead>
</table>

### Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 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>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

**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

**public**

```cpp
InputHandler ()
   constructor
```

```cpp
virtual ~InputHandler ()
   destructor
```

```cpp
bool IsAttached () const
   return true if the input handler is currently attached
```

```cpp
virtual void BeginCapture ()
   capture input to this event handler
```

```cpp
virtual void EndCapture ()
   end input capturing to this event handler
```

```cpp
bool IsCapturing () const
   return true if this input handler captures input
```

```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)
   return true if this object is instance of given class by string
```

```cpp
bool IsInstanceOf (const Util::FourCC &classFourCC)
   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
```
<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</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 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 handler looses capture</td>
</tr>
<tr>
<td>virtual bool</td>
<td><code>OnEvent(const InputEvent &amp;inputEvent)</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]
```

code
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]
```

code
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 Tue Feb 19 12:16:47 2008
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
Input::InputServer Class Reference

#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

<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>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></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 (up to 4)</td>
</tr>
<tr>
<td><code>AttachInputHandler(Input::InputPriority::Code pri, const Ptr&lt;Input::InputHandler&gt; &amp;inputHandler)</code></td>
<td>attach an input handler</td>
</tr>
<tr>
<td><code>RemoveInputHandler(const Ptr&lt;Input::InputHandler&gt; &amp;inputHandler)</code></td>
<td>remove an input handler</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>BeginFrame()</code></td>
<td>call before processing window events</td>
</tr>
<tr>
<td><code>EndFrame()</code></td>
<td>call at end of frame</td>
</tr>
<tr>
<td><code>PutEvent(const Input::InputEvent &amp;ie)</code></td>
<td>put an input event into the handler chain</td>
</tr>
<tr>
<td><code>ClearMouseCapture()</code></td>
<td>clear the current mouse capture (if exists)</td>
</tr>
<tr>
<td><code>ClearKeyboardCapture()</code></td>
<td>clear the current keyboard capture (if exists)</td>
</tr>
<tr>
<td><code>ClearCapture()</code></td>
<td>clear both mouse and keyboard captures</td>
</tr>
<tr>
<td><code>GetMouseCaptureHandler()</code> (const &amp;<a href="">Input::InputHandler</a>) &amp;</td>
<td>return the current mouse capture input handler (return invalid ptr if no capture set)</td>
</tr>
<tr>
<td><code>GetKeyboardCaptureHandler()</code> (const &amp;<a href="">Input::InputHandler</a>) &amp;</td>
<td>return the current keyboard capture input handler (return invalid ptr if no capture set)</td>
</tr>
<tr>
<td><code>ObtainMouseCapture(const &amp;&lt;Input::InputHandler&gt;)</code> &amp;</td>
<td>only call from InputHandler: capture mouse input to the given input handler</td>
</tr>
<tr>
<td><code>ReleaseMouseCapture(const &amp;&lt;Input::InputHandler&gt;)</code> &amp;</td>
<td>only call from InputHandler: release mouse capture</td>
</tr>
<tr>
<td><code>ObtainKeyboardCapture(const &amp;&lt;Input::InputHandler&gt;)</code> &amp;</td>
<td>only call from InputHandler: capture keyboard input to the given input handler</td>
</tr>
<tr>
<td><code>ReleaseKeyboardCapture(const &amp;&lt;Input::InputHandler&gt;)</code> &amp;</td>
<td>only call from InputHandler: release keyboard capture</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><strong>void</strong></td>
<td><strong>Release ()</strong></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><em>decrement refcount and destroy object if refcount is zero</em></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

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

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC () const</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>get the class FourCC code</em></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!)
```
**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; GetMouseMove () const</code></td>
<td>get the current mouse movement</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
bool Win32::Win32InputServer::OpenDInputMouse() [protected, inherited]
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).

```cpp
void Win32::Win32InputServer::CloseDInputMouse() [protected, inherited]
shutdown the DirectInput mouse device
Close the DirectInput mouse and DirectInput.

```cpp
void Win32::Win32InputServer::ReadDInputMouse() [protected, inherited]
get mouse readings
Read data from the DirectInput mouse (relative mouse movement since the last frame).

```cpp
void Base::InputServerBase::EndFrame() [inherited]
call at end of frame
Call this somewhere towards the end of frame, when it is guaraneteed that noone needs input anymore.

```cpp
void Base::InputServerBase::PutEvent(const Input::InputEvent &ie) [inherited]
pull an input event into the handler chain
NOTE: MouseMove and RawMouseMove events will be distributed to all input handlers regardless of mouse capture state!
clear the current mouse capture (if exists)
This clears the currently set mouse capture (if exists).

clear the current keyboard capture (if exists)
This clears the currently set keyboard capture (if exists).

clear both mouse and keyboard captures
This clears the mouse and keyboards captures, if set.

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.

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 `ReleaseKeyboardCapture()` is called.

```cpp
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.

```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

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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>key 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 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>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:47 2008
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>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void BeginCapture</td>
<td><code>virtual void BeginCapture ()</code></td>
<td>capture input to this event handler</td>
</tr>
<tr>
<td>virtual void EndCapture</td>
<td><code>virtual void EndCapture ()</code></td>
<td>end input capturing to this event handler</td>
</tr>
<tr>
<td>bool KeyPressed</td>
<td><code>bool KeyPressed (Input::Key::Code keyCode) const</code></td>
<td>return true if a key is currently pressed</td>
</tr>
<tr>
<td>bool KeyDown</td>
<td><code>bool KeyDown (Input::Key::Code keyCode) const</code></td>
<td>return true if key was down at least once in current frame</td>
</tr>
<tr>
<td>bool KeyUp</td>
<td><code>bool KeyUp (Input::Key::Code keyCode) const</code></td>
<td>return true if key was up at least once in current frame</td>
</tr>
<tr>
<td>const Util::String &amp; GetCharInput</td>
<td><code>const Util::String &amp; GetCharInput () const</code></td>
<td>get character input in current frame</td>
</tr>
<tr>
<td>bool IsAttached</td>
<td><code>bool IsAttached () const</code></td>
<td>return true if the input handler is currently attached</td>
</tr>
<tr>
<td>bool IsCapturing</td>
<td><code>bool IsCapturing () const</code></td>
<td>return true if this input handler captures input</td>
</tr>
<tr>
<td>int GetRefCount</td>
<td><code>int GetRefCount () const</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef</td>
<td><code>void AddRef ()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void Release</td>
<td><code>void Release ()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool IsInstanceOf</td>
<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 IsInstanceOf</td>
<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 IsInstanceOf</td>
<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>bool IsA</td>
<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 IsA</td>
<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>
</tbody>
</table>
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><em>dump refcounting leaks, call at end of application (NEBUŁA3_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 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</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**

```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.
Input: Mouse
#include <mouse.h>

Inheritance diagram for Input::Mouse:
Detailed Description

An input handler which represents a mouse for polling.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const Math::float2 &amp; GetMovement () const</code></td>
<td>Get mouse movement</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 ButtonPressed (Input::MouseButton::Code btn) const</code></td>
<td>Return true if button is currently pressed</td>
</tr>
<tr>
<td><code>bool ButtonDown (Input::MouseButton::Code btn) const</code></td>
<td>Return true if button was down at least once in current frame</td>
</tr>
<tr>
<td><code>bool ButtonUp (Input::MouseButton::Code btn) const</code></td>
<td>Return true if button was up at least once in current frame</td>
</tr>
<tr>
<td><code>bool ButtonDoubleClicked (Input::MouseButton::Code btn) const</code></td>
<td>Return true if a button has been double clicked</td>
</tr>
<tr>
<td><code>bool WheelForward () const</code></td>
<td>Return true if mouse wheel rotated forward</td>
</tr>
<tr>
<td><code>bool WheelBackward () const</code></td>
<td>Return true if mouse wheel rotated backward</td>
</tr>
<tr>
<td><code>const Math::float2 &amp; GetPixelPosition () const</code></td>
<td>Get current absolute mouse position (in pixels)</td>
</tr>
<tr>
<td><code>const Math::float2 &amp; GetScreenPosition () const</code></td>
<td>Get current screen space mouse position (0.0 .. 1.0)</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>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>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 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

```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:: 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>static Util::String</th>
<th><strong>ToString</strong> <em>(Code code)</em></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> <em>(const Util::String &amp;str)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>convert from string</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:48 2008
Interface::CopyFile
#include <copyfile.h>

Inheritance diagram for Interface::CopyFile:

```
Core::RefCounted
    ↓
Messaging::Message
    ↓
Interface::CopyFile
```
Detailed Description

IOInterface message to asynchronously copy a file.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CopyFile ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>SetFromURI (const IO::URI &amp;uri)</strong></td>
<td>set origin URI</td>
</tr>
<tr>
<td><strong>GetFromURI () const</strong></td>
<td>get origin URI</td>
</tr>
<tr>
<td><strong>SetToURI (const IO::URI &amp;uri)</strong></td>
<td>set destination URI</td>
</tr>
<tr>
<td><strong>GetToURI () const</strong></td>
<td>get destination URI</td>
</tr>
<tr>
<td><strong>SetResult (bool b)</strong></td>
<td>set result of operation (true is success)</td>
</tr>
<tr>
<td><strong>GetResult () const</strong></td>
<td>get result of operation</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>CheckId (const Messaging::Id &amp;id) const</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>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>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Rtti &amp;rtti) const</code></td>
<td>Returns true if this object is instance of given class</td>
</tr>
<tr>
<td><code>IsInstanceOf (const Util::String &amp;className) const</code></td>
<td>Returns 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>Returns 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>Returns 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>Returns 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>Returns 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

```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.
Interface::CreateDirectory
Interface::CreateDirectory Class Reference

#include <createdirectory.h>

Inheritance diagram for Interface::CreateDirectory:
Detailed Description

IOInterface message to create a filesystem directory.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetURI (const IO::URI &amp;uri)</code></td>
<td>void in: set URI of new directory</td>
</tr>
<tr>
<td><code>GetURI () const</code></td>
<td>const IO::URI &amp; in: get URI of new directory</td>
</tr>
<tr>
<td><code>SetResult (bool b)</code></td>
<td>void set result of operation (true is success)</td>
</tr>
<tr>
<td><code>GetResult () const</code></td>
<td>bool get result of operation</td>
</tr>
<tr>
<td><code>Encode (const Ptr&lt; IO::BinaryWriter &gt; &amp;writer)</code></td>
<td>virtual void encode message into a stream</td>
</tr>
<tr>
<td><code>Decode (const Ptr&lt; IO::BinaryReader &gt; &amp;reader)</code></td>
<td>virtual void decode message from a stream</td>
</tr>
<tr>
<td><code>CheckId (const Messaging::Id &amp;id) const</code></td>
<td>bool return true if message is of the given id</td>
</tr>
<tr>
<td><code>SetHandled (bool b)</code></td>
<td>void set the handled flag</td>
</tr>
<tr>
<td><code>Handled () const</code></td>
<td>bool return true if the message has been handled</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)</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 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>
<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.
Interface::DeleteDirectory
# Interface::DeleteDirectory Class Reference

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

Inheritance diagram for Interface::DeleteDirectory:
Detailed Description

IOInterface message to delete a directory.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetURI (const IO::URI &amp;uri)</code></td>
<td>set URI of new directory</td>
</tr>
<tr>
<td><code>const IO::URI &amp; GetURI () const</code></td>
<td>get URI of new directory</td>
</tr>
<tr>
<td><code>void SetResult (bool b)</code></td>
<td>set result of operation (true is success)</td>
</tr>
<tr>
<td><code>bool GetResult () const</code></td>
<td>get result of operation</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>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>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>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>
</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
<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.
Interface::DeleteFile
#include <deletefile.h>

Inheritance diagram for Interface::DeleteFile:
Detailed Description

IOInterface message to asynchronously delete a file.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void SetURI (const IO::URI &amp;uri)</td>
<td><em>in:</em> set URI of new directory</td>
</tr>
<tr>
<td>const IO::URI &amp; GetURI () const</td>
<td><em>in:</em> get URI of new directory</td>
</tr>
<tr>
<td>void SetResult (bool b)</td>
<td>set result of operation (true is success)</td>
</tr>
<tr>
<td>bool GetResult () const</td>
<td>get result of operation</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>bool CheckId (const Messaging::Id &amp;id) const</td>
<td>return true if message is of the given id</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>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>Function</td>
<td>Signature</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>(const <code>Util::FourCC</code> &amp;classFourCC) const</td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>(const <code>Rtti</code> &amp;rtti) const</td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>(const <code>Util::String</code> &amp;rttiName) const</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><code>IsA</code></td>
<td>(const <code>Util::FourCC</code> &amp;rttiFourCC) const</td>
</tr>
<tr>
<td></td>
<td></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 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.
Interface::IOMessage
# Interface::IOMessage Class Reference

#include <iomessage.h>

Inheritance diagram for Interface::IOMessage:
Detailed Description

**Base** class for IOInterface messages.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IOMessage ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>void SetURI (const IO::URI &amp;uri)</code></td>
<td>set URI of new directory</td>
</tr>
<tr>
<td><code>const IO::URI &amp; GetURI () const</code></td>
<td>get URI of new directory</td>
</tr>
<tr>
<td><code>void SetResult (bool b)</code></td>
<td>set result of operation (true is success)</td>
</tr>
<tr>
<td><code>bool GetResult () const</code></td>
<td>get result of operation</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>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>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</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.
Interface::MountZipArchive
#include <mountziparchive.h>

Inheritance diagram for Interface::MountZipArchive:
Detailed Description

IOInterface message to mount a zip archive.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetURI (const IO::URI &amp;uri)</code></td>
<td>in: set URI of new directory</td>
</tr>
<tr>
<td><code>const IO::URI &amp; GetURI () const</code></td>
<td>get URI of new directory</td>
</tr>
<tr>
<td><code>void SetResult (bool b)</code></td>
<td>set result of operation (true is success)</td>
</tr>
<tr>
<td><code>bool GetResult () const</code></td>
<td>get result of operation</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>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>void SetHandled (bool b)</code></td>
<td>set the handled flag</td>
</tr>
<tr>
<td><code>bool Helded () const</code></td>
<td>return true if the message has been handled</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><code>IsInstanceOf</code> (const <code>Util::FourCC</code> &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><code>IsA</code> (const <code>Rtti</code> &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><code>IsA</code> (const <code>Util::String</code> &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><code>IsA</code> (const <code>Util::FourCC</code> &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 <code>Util::String</code> &amp;</th>
<th><code>GetClassName</code> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>get the class name</em></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><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></td>
<td><code>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</code></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.
Interface::ReadStream
# Interface::ReadStream Class Reference

```
#include <readstream.h>
```

Inheritance diagram for Interface::ReadStream:
Detailed Description

IOInterface message to asynchronously read a stream. Note that the caller must provide a mappable stream which consumes the read content.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetStream</strong> (const Ptr&lt; IO::Stream &gt; &amp;stream)</td>
<td>set a stream which consumes the result of the read operation</td>
</tr>
<tr>
<td><strong>GetStream</strong> () const</td>
<td>get stream with read result (after message has been handled!)</td>
</tr>
<tr>
<td><strong>Encode</strong> (const Ptr&lt; IO::BinaryWriter &gt; &amp;writer)</td>
<td>encode message into a stream</td>
</tr>
<tr>
<td><strong>Decode</strong> (const Ptr&lt; IO::BinaryReader &gt; &amp;reader)</td>
<td>decode message from a stream</td>
</tr>
<tr>
<td><strong>SetURI</strong> (const IO::URI &amp;uri)</td>
<td>in: set URI of new directory</td>
</tr>
<tr>
<td><strong>GetURI</strong> () const</td>
<td>in: get URI of new directory</td>
</tr>
<tr>
<td><strong>SetResult</strong> (bool b)</td>
<td>set result of operation (true is success)</td>
</tr>
<tr>
<td><strong>GetResult</strong> () const</td>
<td>get result of operation</td>
</tr>
<tr>
<td><strong>CheckId</strong> (const Messaging::Id &amp;id) const</td>
<td>return true if message is of the given id</td>
</tr>
<tr>
<td><strong>SetHandled</strong> (bool b)</td>
<td>set the handled flag</td>
</tr>
<tr>
<td><strong>Handled</strong> () const</td>
<td>return true if the message has been handled</td>
</tr>
<tr>
<td><strong>GetRefCount</strong> () const</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><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 DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
set a stream which consumes the result of the read operation

Set the caller-provided stream which will be filled with the read data. Note that the contents of the stream will only be valid after the message has been handled by IOInterface.

get stream with read result (after message has been handled!)

Get the stream which will contain the read data (after the message has been handled).

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.
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::WriteStream
#include <writestream.h>

Inheritance diagram for Interface::WriteStream:
Detailed Description

IOInterface message to asynchronously write a stream.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void SetStream (const Ptr&lt; IO::Stream &gt; &amp;stream)</td>
<td>set stream which provides the data to write</td>
</tr>
<tr>
<td>const Ptr&lt; IO::Stream &gt; &amp; GetStream () const</td>
<td>get stream which provides the data</td>
</tr>
<tr>
<td>void Encode (const Ptr&lt; IO::BinaryWriter &gt; &amp;writer)</td>
<td>encode message into a stream</td>
</tr>
<tr>
<td>void Decode (const Ptr&lt; IO::BinaryReader &gt; &amp;reader)</td>
<td>decode message from a stream</td>
</tr>
<tr>
<td>void SetURI (const IO::URI &amp;uri)</td>
<td>in: set URI of new directory</td>
</tr>
<tr>
<td>const IO::URI &amp; GetURI () const</td>
<td>in: get URI of new directory</td>
</tr>
<tr>
<td>void SetResult (bool b)</td>
<td>set result of operation (true is success)</td>
</tr>
<tr>
<td>bool GetResult () const</td>
<td>get result of operation</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>bool CheckId (const Messaging::Id &amp;id) const</td>
<td>return true if message is of the given id</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>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</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>Increment refcount by one, 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><code>DumpRefCountingLeaks</code> ()</th>
</tr>
</thead>
</table>

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

```c++
void Interface::WriteStream::SetStream(const Ptr<IO::Stream> & s) [inline]
```

set stream which provides the data to write

NOTE: We don't make a copy of the stream. Streams are multithreading-safe in a way that only one thread may open a stream at any time. If this introduces multithreading issues it's safest to make a copy of the stream.

```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.
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.
Internal::FSWrapper
Internal::FSWrapper Class Reference

#include <fswrapper.h>

Inheritance diagram for Internal::FSWrapper:

```
Win32::Win32FSWrapper

Internal::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
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OpenFile</code></td>
<td>(const <code>Util::String</code> &amp;path, <code>IO::Stream::AccessMode</code> accessMode, <code>IO::Stream::AccessPattern</code> accessPattern) open a file</td>
</tr>
<tr>
<td><code>CloseFile</code></td>
<td>(Handle h) close a file</td>
</tr>
<tr>
<td><code>Write</code></td>
<td>(Handle h, const void *buf, <code>IO::Stream::Size</code> numBytes) write to a file</td>
</tr>
<tr>
<td><code>Read</code></td>
<td>(Handle h, void *buf, <code>IO::Stream::Size</code> numBytes) read from a file</td>
</tr>
<tr>
<td><code>Seek</code></td>
<td>(Handle h, <code>IO::Stream::Offset</code> offset, <code>IO::Stream::SeekOrigin</code> orig) seek in a file</td>
</tr>
<tr>
<td><code>Tell</code></td>
<td>(Handle h) get position in file</td>
</tr>
<tr>
<td><code>Flush</code></td>
<td>(Handle h) flush a file</td>
</tr>
<tr>
<td><code>Eof</code></td>
<td>(Handle h) return true if at end-of-file</td>
</tr>
<tr>
<td><code>GetFileSize</code></td>
<td>(Handle h) get size of a file in bytes</td>
</tr>
<tr>
<td><code>SetReadOnly</code></td>
<td>(const <code>Util::String</code> &amp;path, bool readOnly) set read-only status of a file</td>
</tr>
<tr>
<td><code>IsReadOnly</code></td>
<td>(const <code>Util::String</code> &amp;path) get read-only status of a file</td>
</tr>
<tr>
<td><code>DeleteFile</code></td>
<td>(const <code>Util::String</code> &amp;path) delete a file</td>
</tr>
<tr>
<td><code>DeleteDirectory</code></td>
<td>(const <code>Util::String</code> &amp;path) delete an empty directory</td>
</tr>
<tr>
<td><code>FileExists</code></td>
<td>(const <code>Util::String</code> &amp;path)</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>return true if a file exists</code></td>
<td>DirectoryExists</td>
</tr>
<tr>
<td><code>return true if a directory exists</code></td>
<td>DirectoryExists</td>
</tr>
<tr>
<td><code>get the last write-access timestamp of a file</code></td>
<td>GetWriteTime</td>
</tr>
<tr>
<td><code>create a directory</code></td>
<td>CreateDirectory</td>
</tr>
<tr>
<td><code>list all files in a directory</code></td>
<td>ListFiles</td>
</tr>
<tr>
<td><code>list all subdirectories in a directory</code></td>
<td>ListDirectories</td>
</tr>
<tr>
<td><code>get path to the current user’s home directory (for user: standard assign)</code></td>
<td>GetUserDirectory</td>
</tr>
<tr>
<td><code>get path to the current user’s temp directory (for temp: standard assign)</code></td>
<td>GetTempDirectory</td>
</tr>
<tr>
<td><code>get path to the current application directory (for home: standard assign)</code></td>
<td>GetHomeDirectory</td>
</tr>
<tr>
<td><code>get path to the current bin directory (for bin: standard assign)</code></td>
<td>GetBinDirectory</td>
</tr>
<tr>
<td><code>return true when the string is a device name (e.g. &quot;C:&quot;</code>)</td>
<td>IsDeviceName</td>
</tr>
</tbody>
</table>
Member Function Documentation

Win32FSWrapper::Handle
Win32::Win32FSWrapper::OpenFile (const Util::String & path,
    IO::Stream::AccessMode accessMode,
    IO::Stream::AccessPattern accessPattern ) [static, inherited]

open a file

Open a file using the **Win32** function CreateFile(). Returns a handle to
the file which must be passed to the other **Win32FSWrapper** file
methods. If opening the file fails, the function will return 0. The
filename must be a native **Win32** path (no assigns, etc...).

void
Win32::Win32FSWrapper::CloseFile (Handle handle ) [static, inherited]

close a file

Closes a file opened by **Win32FSWrapper::OpenFile()**.

Stream::Position
Win32::Win32FSWrapper::Tell (Handle handle ) [static, inherited]

get position in file

Get current position in file.

void
Win32::Win32FSWrapper::Flush (Handle handle ) [static, inherited]

flush a file

Flush unwritten data to file.

bool
Win32::Win32FSWrapper::Eof (Handle handle ) [static, inherited]

return true if at end-of-file
Returns true if current position is at end of file.

Stream::Size
Win32::Win32FSWrapper::GetFileSize (Handle handle) [static, inherited]

get size of a file in bytes

Returns the size of a file in bytes.

void Win32::Win32FSWrapper::SetReadOnly (const Util::String &path, bool readOnly) [static, inherited]

set read-only status of a file

Set the read-only status of a file.

bool Win32::Win32FSWrapper::IsReadOnly (const Util::String &path) [static, inherited]

get read-only status of a file

Get the read-only status of a file.

bool Win32::Win32FSWrapper::DeleteFile (const Util::String &path) [static, inherited]

delete a file

Deletes a file. Returns true if the operation was successful. The delete will fail if the fail doesn't exist or the file is read-only.

bool Win32::Win32FSWrapper::DeleteDirectory (const Util::String &path) [static, inherited]

delete an empty directory

Delete an empty directory. Returns true if the operation was
successful.

```cpp
bool Win32::Win32FSWrapper::FileExists(const Util::String path ) [static, inherited]
return true if a file exists
Return true if a file exists.
```

```cpp
bool Win32::Win32FSWrapper::DirectoryExists(const Util::String path ) [static, inherited]
return true if a directory exists
Return true if a directory exists.
```

```cpp
FileTime Win32::Win32FSWrapper::GetFileWriteTime(const Util::String path ) [static, inherited]
get the last write-access time stamp of a file
Return the last write-access time to a file.
```

```cpp
bool Win32::Win32FSWrapper::CreateDirectory(const Util::String path ) [static, inherited]
create a directory
Creates a new directory.
```

```cpp
Array< String > Win32::Win32FSWrapper::ListFiles(const Util::String dirPath, const Util::String pattern & ) [static, inherited]
list all files in a directory
```
Lists all files in a directory, filtered by a pattern.

```
Array<String>
Win32::Win32FSWrapper::ListDirectories()
```

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
Win32::Win32FSWrapper::GetUserDirectory()
```

get path to the current user's home directory (for user: standard assign)

This method should return the path to the current user's home directory. This is the directory where application can write their data to. Under windows, this is the "My Files" directory.

```
String
Win32::Win32FSWrapper::GetTempDirectory()
```

get path to the current user's temp directory (for temp: standard assign)

This method should return a directory for temporary files with read/write access for the current user.

```
String
Win32::Win32FSWrapper::GetHomeDirectory()
```

get path to the current application directory (for home: standard assign)

This method should return the installation directory of the application. Under Nebula3, this is either the directory where the executable is
located, or 2 levels above the executable (if it is in home:bin/win32).

String
Win32::Win32FSWrapper::GetBinDirectory ( ) [static, inherited]

get path to the current bin directory (for bin: standard assign)

This method sould return the directory where the application executable is located.
IO::Assign
IO::Assign Class Reference

#include <assign.h>

Inheritance diagram for IO::Assign:

```
Util::KeyValuePair< Util::String, Util::String >
```

```
< 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>default constructor</td>
</tr>
</tbody>
</table>
|                   | **Assign (const **Util::String** &name, const **Util::String** &path)**  
|                   | constructor with assign name and path            |
| const **Util::String** & **GetName ()** const | get name of assign                               |
| const **Util::String** & **GetPath ()** const   | get path of assign                               |
| **operator==**    | (const **KeyValuePair**< **Util::String**, **Util::String** > &rhs) const  
|                   | equality operator                                |
| **operator!=**    | (const **KeyValuePair**< **Util::String**, **Util::String** > &rhs) const  
|                   | inequality operator                              |
| **operator>**     | (const **KeyValuePair**< **Util::String**, **Util::String** > &rhs) const  
|                   | greater operator                                 |
| **operator>=**    | (const **KeyValuePair**< **Util::String**, **Util::String** > &rhs) const  
|                   | greater-or-equal operator                        |
| **operator<**     | (const **KeyValuePair**< **Util::String**, **Util::String** > &rhs) const  
|                   | lesser operator                                  |
| **operator<==**   | (const **KeyValuePair**< **Util::String**, **Util::String** > &rhs) const  
|                   | lesser-or-equal operator                         |
| **value ()**      | read/write access to value                       |
| const **Util::String** & **Value ()** const     | read access to key                               |
| const **Util::String** & **Key ()** const       | read access to key                               |
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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinaryReader ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~BinaryReader ()</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>char ReadChar ()</td>
<td>read an 8-bit char from the stream</td>
</tr>
<tr>
<td>unsigned char ReadUChar ()</td>
<td>read an 8-bit unsigned character from the stream</td>
</tr>
<tr>
<td>short ReadShort ()</td>
<td>read a 16-bit short from the stream</td>
</tr>
<tr>
<td>unsigned short ReadUShort ()</td>
<td>read a 16-bit unsigned short from the stream</td>
</tr>
<tr>
<td>int ReadInt ()</td>
<td>read a 32-bit int from the stream</td>
</tr>
<tr>
<td>unsigned int ReadUInt ()</td>
<td>read a 32-bit unsigned int from the stream</td>
</tr>
<tr>
<td>float ReadFloat ()</td>
<td>read a float value from the stream</td>
</tr>
<tr>
<td>double ReadDouble ()</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>bool</td>
<td>ReadBool ()</td>
</tr>
<tr>
<td>Util::String</td>
<td>ReadString ()</td>
</tr>
<tr>
<td>Math::float4</td>
<td>ReadFloat4 ()</td>
</tr>
<tr>
<td>Math::matrix44</td>
<td>ReadMatrix44 ()</td>
</tr>
<tr>
<td>Util::Blob</td>
<td>ReadBlob ()</td>
</tr>
<tr>
<td>Util::Guid</td>
<td>ReadGuid ()</td>
</tr>
<tr>
<td>void</td>
<td>SetStream (const Ptr&lt; Stream &gt; &amp;s)</td>
</tr>
<tr>
<td>const Ptr&lt; Stream &gt; &amp;</td>
<td>GetStream () const</td>
</tr>
<tr>
<td>bool</td>
<td>HasStream () const</td>
</tr>
<tr>
<td>bool</td>
<td>Eof () const</td>
</tr>
<tr>
<td>bool</td>
<td>IsOpen () const</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>bool</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
</tr>
</tbody>
</table>
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><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

```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:

- **Core::RefCounted**
- **IO::StreamWriter**
- **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 <strong>SetMemoryMappingEnabled</strong> (bool b)</td>
<td>call before <strong>Open()</strong> to enable memory mapping (if stream supports mapping)</td>
</tr>
<tr>
<td>bool <strong>IsMemoryMappingEnabled</strong> () const</td>
<td>return true if memory mapping is enabled</td>
</tr>
<tr>
<td>void <strong>SetStreamByteOrder</strong> (System::ByteOrder::Type byteOrder)</td>
<td>set the stream byte order (default is host byte order)</td>
</tr>
<tr>
<td>System::ByteOrder::Type <strong>GetStreamByteOrder</strong> () const</td>
<td>get the stream byte order</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>void <strong>WriteChar</strong> (char c)</td>
<td>write an 8-bit char to the stream</td>
</tr>
<tr>
<td>void <strong>WriteUChar</strong> (unsigned char c)</td>
<td>write an 8-bit unsigned char to the stream</td>
</tr>
<tr>
<td>void <strong>WriteShort</strong> (short s)</td>
<td>write an 16-bit short to the stream</td>
</tr>
<tr>
<td>void <strong>WriteUShort</strong> (unsigned short s)</td>
<td>write an 16-bit unsigned short to the stream</td>
</tr>
<tr>
<td>void <strong>WriteInt</strong> (int i)</td>
<td>write an 32-bit int to the stream</td>
</tr>
<tr>
<td>void <strong>WriteUInt</strong> (unsigned int i)</td>
<td>write an 32-bit unsigned int to the stream</td>
</tr>
<tr>
<td>void <strong>WriteFloat</strong> (float f)</td>
<td>write a float value to the stream</td>
</tr>
<tr>
<td>void <strong>WriteDouble</strong> (double d)</td>
<td></td>
</tr>
</tbody>
</table>
void WriteBool (bool b)
write a boolean value to the stream

void WriteString (const Util::String &s)
write a string to the stream

void WriteFloat4 (const Math::float4 &v)
write a float4 to the stream

void WriteMatrix44 (const Math::matrix44 &m)
write a matrix44 to the stream

void WriteBlob (const Util::Blob &blob)
write a blob of data

void WriteGuid (const Util::Guid &guid)
write a guid

void SetStream (const Ptr<Stream> &s)
set stream to write to

const Ptr<Stream> & GetStream () const
generate 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

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

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.

```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::Console
#include <console.h>

Inheritance diagram for IO::Console:

```
Core::RefCounted
         ^
         |  
IO::Console
```
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><code>(Console)</code></td>
<td>virtual constructor</td>
</tr>
<tr>
<td><code>(~Console)</code></td>
<td>virtual destructor</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>void open the console</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>void close the console</td>
</tr>
<tr>
<td><code>IsOpen()</code></td>
<td>bool return true if currently open</td>
</tr>
<tr>
<td><code>Update()</code></td>
<td>void called per-frame</td>
</tr>
<tr>
<td><code>AttachHandler()</code></td>
<td>void attach a console handler to the console</td>
</tr>
<tr>
<td><code>RemoveHandler()</code></td>
<td>void remove a console handler from the console</td>
</tr>
<tr>
<td><code>GetNumHandlers()</code></td>
<td>SizeT get number of attached console handlers</td>
</tr>
<tr>
<td><code>GetHandlerByIndex()</code></td>
<td>const get attached console handler at index</td>
</tr>
<tr>
<td><code>HasInput()</code></td>
<td>bool return true if user input is available</td>
</tr>
<tr>
<td><code>GetInput()</code></td>
<td>const get user input</td>
</tr>
<tr>
<td><code>Print()</code></td>
<td>void print a formatted line (printf style)</td>
</tr>
<tr>
<td><code>Print()</code></td>
<td>void print a formatted line (printf style) with va_list argument</td>
</tr>
<tr>
<td><code>Print()</code></td>
<td>void print a string object</td>
</tr>
<tr>
<td><code>Error()</code></td>
<td>void print a formatted line (printf style) with va_list argument</td>
</tr>
<tr>
<td><code>Util::String GetInput()</code></td>
<td>const get user input</td>
</tr>
<tr>
<td><code>Util::String GetInput()</code></td>
<td>const get user input</td>
</tr>
<tr>
<td><code>Util::String GetInput()</code></td>
<td>const get user input</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>Error</code> (const char *fmt, va_list argList)</td>
<td>put an error message and cancel execution</td>
</tr>
<tr>
<td><code>Warning</code> (const char *fmt,...)</td>
<td>put a warning message</td>
</tr>
<tr>
<td><code>DebugOut</code> (const char *fmt,...)</td>
<td>print a debug-only message</td>
</tr>
<tr>
<td><code>DebugOut</code> (const char *fmt, va_list argList)</td>
<td>print a debug-only 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</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>Function Type</th>
<th>Function Name</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.

```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::ConsoleHandler
#include <consolehandler.h>

Inheritance diagram for IO::ConsoleHandler:

```
Core::RefCounted
  |
  ↓
IO::ConsoleHandler
  |
  ↓
Win32::Win32ConsoleHandler
```
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>Constructor</td>
</tr>
<tr>
<td>virtual ~ConsoleHandler ()</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>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 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>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>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool 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>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>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.

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.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

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>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><strong>FileStream ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~FileStream ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>CanRead ()</strong> const</td>
<td>supports reading</td>
</tr>
<tr>
<td><strong>CanWrite ()</strong> const</td>
<td>supports writing</td>
</tr>
<tr>
<td><strong>CanSeek ()</strong> const</td>
<td>supports seeking</td>
</tr>
<tr>
<td><strong>CanBeMapped ()</strong> const</td>
<td>supports memory mapping (read-only)</td>
</tr>
<tr>
<td><strong>GetSize ()</strong> const</td>
<td>get the size of the stream in bytes</td>
</tr>
<tr>
<td><strong>GetPosition ()</strong> const</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 ()</strong> const</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

custom 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>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><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

bool IO::FileStream::CanBeMapped() const [virtual]
supports memory mapping (read-only)
FileStream 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 is smaller than 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 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(const void* ptr, Size numBytes)
)
```

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}
Stream::Size 
IO::Stream::Read ( * ptr, 
    Size numBytes
 ) [virtual, inherited]
\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 then numBytes, or 0 if end-of-stream is reached.

\begin{verbatim}
void 
IO::Stream::Seek ( Offset offset, 
    SeekOrigin origin 
 ) [virtual, inherited]
\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}
bool 
IO::Stream::IsMapped ( ) const [inherited]
\end{verbatim}

return true if stream is currently mapped to memory

Returns true if the stream is currently mapped.

\begin{verbatim}
int 
Core::RefCounted::GetRefCount ( ) const [inline, inherited]
\end{verbatim}

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>

Inheritance diagram for IO::FileTime:

```
Win32::Win32FileTime

IO::FileTime
```

Detailed Description

Defines a file-access timestamp.

(C) 2006 Radon Labs GmbH
**Friends**

<table>
<thead>
<tr>
<th>bool</th>
<th>operator == (const Win32FileTime &amp;a, const Win32FileTime &amp;b)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>operator ==</td>
</tr>
<tr>
<td>bool</td>
<td>operator != (const Win32FileTime &amp;a, const Win32FileTime &amp;b)</td>
</tr>
<tr>
<td></td>
<td>operator !=</td>
</tr>
<tr>
<td>bool</td>
<td>operator &gt; (const Win32FileTime &amp;a, const Win32FileTime &amp;b)</td>
</tr>
<tr>
<td></td>
<td>operator &gt;</td>
</tr>
<tr>
<td>bool</td>
<td>operator &lt; (const Win32FileTime &amp;a, const Win32FileTime &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

**IO::Interface**
IO::Interface Class Reference

#include <iointerfacehandler.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

Platform dependent handler for io interface. Needs to register the stream schemes at io server.

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

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::I0InterfaceHandlerBase
IO::IOInterfaceHandlerBase Class Reference

#include <iointerfacehandlerbase.h>
Detailed Description

Handler class for io interfaces.

(C) 2006 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::IoServer
#include <ioserver.h>

Inheritance diagram for 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>void 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>void UnregisterUriScheme (const Util::String &amp;uriScheme)</strong></td>
<td>unregister an uri scheme</td>
</tr>
<tr>
<td><strong>bool IsUriSchemeRegistered (const Util::String &amp;uriScheme) const</strong></td>
<td>return true if an uri scheme has been registered</td>
</tr>
<tr>
<td><strong>const Core::Rtti &amp; GetStreamClassByUriScheme (const Util::String &amp;uriScheme) const</strong></td>
<td>get the registered stream class for an uri scheme</td>
</tr>
<tr>
<td><strong>Util::Array &lt; Util::String &gt; GetAllRegisteredUriSchemes () const</strong></td>
<td>get an array of all registered schemes</td>
</tr>
<tr>
<td><strong>Ptr&lt; Stream &gt; CreateStream (const URI &amp;uri) const</strong></td>
<td>create a stream object for the given uri</td>
</tr>
<tr>
<td><strong>bool MountZipArchive (const URI &amp;uri)</strong></td>
<td>mount a zip file archive</td>
</tr>
<tr>
<td><strong>void UnmountZipArchive (const URI &amp;uri)</strong></td>
<td>unmount a zip file archive</td>
</tr>
<tr>
<td><strong>bool IsZipArchiveMounted (const URI &amp;uri) const</strong></td>
<td>return true if a zip archive is mounted</td>
</tr>
<tr>
<td><strong>void SetZipFileSystemEnabled (bool b)</strong></td>
<td>enable/disable transparent zip filesystem layering (default is yes)</td>
</tr>
<tr>
<td><strong>bool IsZipFileSystemEnabled () const</strong></td>
<td>return true if transparent zip filesystem is enabled</td>
</tr>
<tr>
<td><strong>void SetAssign (const Assign &amp;assign)</strong></td>
<td>define a directory assign</td>
</tr>
<tr>
<td><strong>HasAssign (const Util::String &amp;assignName)</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>const</code></td>
</tr>
<tr>
<td><code>GetAssign</code></td>
<td>(const <code>Util::String</code> &amp;assignName) const</td>
</tr>
<tr>
<td><code>ClearAssign</code></td>
<td>(const <code>Util::String</code> &amp;assignName)</td>
</tr>
<tr>
<td><code>GetAllAssigns</code></td>
<td>() const</td>
</tr>
<tr>
<td><code>ResolveAssigns</code></td>
<td>(const <code>URI</code> &amp;uri) const</td>
</tr>
<tr>
<td><code>ResolveAssignsInString</code></td>
<td>(const <code>Util::String</code> &amp;uriString) const</td>
</tr>
<tr>
<td><code>CreateDirectory</code></td>
<td>(const <code>URI</code> &amp;uri) const</td>
</tr>
<tr>
<td><code>DeleteDirectory</code></td>
<td>(const <code>URI</code> &amp;path) const</td>
</tr>
<tr>
<td><code>DirectoryExists</code></td>
<td>(const <code>URI</code> &amp;path) const</td>
</tr>
<tr>
<td><code>CopyFile</code></td>
<td>(const <code>URI</code> &amp;from, const <code>URI</code> &amp;to) const</td>
</tr>
<tr>
<td><code>DeleteFile</code></td>
<td>(const <code>URI</code> &amp;path) const</td>
</tr>
<tr>
<td><code>FileExists</code></td>
<td>(const <code>URI</code> &amp;path) const</td>
</tr>
<tr>
<td><code>SetReadOnly</code></td>
<td>(const <code>URI</code> &amp;path, bool b) const</td>
</tr>
<tr>
<td><code>IsReadOnly</code></td>
<td>(const <code>URI</code> &amp;path) const</td>
</tr>
<tr>
<td><code>ComputeFileCrc</code></td>
<td>(const <code>URI</code> &amp;path) const</td>
</tr>
<tr>
<td><code>GetFileWriteTime</code></td>
<td>(const <code>URI</code> &amp;path) const</td>
</tr>
<tr>
<td><code>ListFiles</code></td>
<td>(const <code>URI</code> &amp;dir, const <code>Util::String</code> &amp;pattern) const</td>
</tr>
</tbody>
</table>

- `bool` `const` return true if an assign exists
- `const Util::String &` get an assign
- `void ClearAssign` (const `Util::String` &assignName) clear an assign
- `Util::Array< Assign >` `GetAllAssigns` () const return an array of all currently defined assigns
- `URI` `ResolveAssigns` (const `URI` &uri) const resolve any assigns in an `URI`
- `Util::String` `ResolveAssignsInString` (const `Util::String` &uriString) const resolve any assigns in a string (must have `URI` form)
- `bool` `CreateDirectory` (const `URI` &uri) const create all missing directories in the path
- `bool` `DeleteDirectory` (const `URI` &path) const delete an empty directory
- `bool` `DirectoryExists` (const `URI` &path) const return true if directory exists
- `bool` `CopyFile` (const `URI` &from, const `URI` &to) const copy a file
- `bool` `DeleteFile` (const `URI` &path) const delete a file
- `bool` `FileExists` (const `URI` &path) const return true if file exists
- `void` `SetReadOnly` (const `URI` &path, bool b) const set the readonly status of a file
- `bool` `IsReadOnly` (const `URI` &path) const return read only status of a file
- `unsigned int` `ComputeFileCrc` (const `URI` &path) const get the CRC checksum of a file
- `FileTime` `GetFileWriteTime` (const `URI` &path) const return the last write-time of a file
- `Util::Array < Util::String >` `ListFiles` (const `URI` &dir, const `Util::String` &pattern) const list all files matching a pattern in a directory
<table>
<thead>
<tr>
<th>Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Util::Array &lt; Util::String &gt;</td>
<td>ListDirectories (const URI &amp;dir, const Util::String &amp;pattern) const</td>
<td>list all subdirectories matching a pattern in a directory</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>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>
void IO::IoServer::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.

**URI**

IO::IoServer::ResolveAssigns(const URI & uri)

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**

IO::IoServer::ResolveAssignsInString(const Util::String uriString &)

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.

**bool**

IO::IoServer::CreateDirectory(const URI & uri)

create all missing directories in the path

This method creates all missing directories in a path.
bool IO::IoServer::CopyFile(const URI & fromUri, const URI & toUri)

This copies a file to another file.

generic int IO::IoServer::ComputeFileCrc(const URI & uri)

get the CRC checksum of a file

This method computes the CRC checksum for a file.

generic int Core::RefCounted::GetRefCount()

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.
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
### Public Member Functions

<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>Constructor from string</td>
</tr>
<tr>
<td><strong>MediaType (const Util::String &amp;type, const Util::String &amp;subType)</strong></td>
<td>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::StringAsString () 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>
The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:48 2008
IO::MemoryStream
#include <memorystream.h>

Inheritance diagram for IO::MemoryStream:

```
IO::MemoryStream
   `-- IO::Stream
       `-- Core::RefCounted
```
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>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>AccessMode</td>
<td>access modes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>enum</td>
<td>AccessPattern</td>
<td>access prefered pattern</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>enum</td>
<td>SeekOrigin</td>
<td>seek origins</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>typedef int</td>
<td>Position</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>MemoryStream ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~MemoryStream ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual bool <strong>CanRead () const</strong></td>
<td>memory streams support reading</td>
</tr>
<tr>
<td>virtual bool <strong>CanWrite () const</strong></td>
<td>memory streams support writing</td>
</tr>
<tr>
<td>virtual bool <strong>CanSeek () const</strong></td>
<td>memory streams support seeking</td>
</tr>
<tr>
<td>virtual bool <strong>CanBeMapped () const</strong></td>
<td>memory streams are mappable</td>
</tr>
<tr>
<td>virtual void <strong>GetSize () const</strong></td>
<td>get the size of the stream in bytes</td>
</tr>
<tr>
<td>virtual void <strong>GetPosition () const</strong></td>
<td>get the current position of the read/write cursor</td>
</tr>
<tr>
<td>virtual void <strong>Open ()</strong></td>
<td>open the stream</td>
</tr>
<tr>
<td>virtual void <strong>Close ()</strong></td>
<td>close the stream</td>
</tr>
<tr>
<td>virtual void *<em>Write (const void <em>ptr, Size numBytes)</em></em></td>
<td>directly write to the stream</td>
</tr>
<tr>
<td>virtual Size *<em>Read (void <em>ptr, Size numBytes)</em></em></td>
<td>directly read from the stream</td>
</tr>
<tr>
<td>virtual void <strong>Seek (Offset offset, SeekOrigin origin)</strong></td>
<td>seek in stream</td>
</tr>
<tr>
<td>virtual bool <strong>Eof () const</strong></td>
<td>return true if end-of-stream reached</td>
</tr>
<tr>
<td>virtual void <strong>Map ()</strong></td>
<td>map for direct memory-access</td>
</tr>
<tr>
<td>virtual void <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

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
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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

static void DumpRefCountingLeaks ()

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

```cpp
Stream::AccessMode 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.

```cpp
Stream::AccessPattern
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
                    ) 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()`.

```cpp
Stream::Size
IO::Stream::Read( const void* ptr,
                    Size numBytes
                    ) const [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
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.

```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]
```

discard 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.
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>enum</th>
<th>AccessMode</th>
<th>access modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>AccessPattern</td>
<td>access preferred pattern</td>
</tr>
<tr>
<td>enum</td>
<td>SeekOrigin</td>
<td>seek origins</td>
</tr>
<tr>
<td>typedef int</td>
<td>Position</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>Stream ()</strong></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td>virtual <strong>~Stream ()</strong></td>
<td><em>destructor</em></td>
</tr>
<tr>
<td>void <strong>SetURI (const URI &amp;u)</strong></td>
<td>set stream location as <em>URI</em></td>
</tr>
<tr>
<td>const URI &amp; <strong>GetURI () const</strong></td>
<td>get stream <em>URI</em></td>
</tr>
<tr>
<td>virtual bool <strong>CanRead () const</strong></td>
<td>return true if the stream supports reading</td>
</tr>
<tr>
<td>virtual bool <strong>CanWrite () const</strong></td>
<td>return true if the stream supports writing</td>
</tr>
<tr>
<td>virtual bool <strong>CanSeek () const</strong></td>
<td>return true if the stream supports seeking</td>
</tr>
<tr>
<td>virtual bool <strong>CanBeMapped () const</strong></td>
<td>return true if the stream provides direct memory access</td>
</tr>
<tr>
<td>virtual void <strong>SetSize (Size s)</strong></td>
<td>set a new size for the stream</td>
</tr>
<tr>
<td>virtual Size <strong>GetSize () const</strong></td>
<td>get the size of the stream in bytes</td>
</tr>
<tr>
<td>virtual Position <strong>GetPosition () const</strong></td>
<td>get the current position of the read/write cursor</td>
</tr>
<tr>
<td>void <strong>SetAccessMode (AccessMode m)</strong></td>
<td>set the access mode of the stream (default is ReadAccess)</td>
</tr>
<tr>
<td><strong>AccessMode</strong> <strong>GetAccessMode () const</strong></td>
<td>get the access mode of the stream</td>
</tr>
<tr>
<td>void <strong>SetAccessPattern (AccessPattern p)</strong></td>
<td>set the preferred access pattern (default is Sequential)</td>
</tr>
<tr>
<td><strong>AccessPattern</strong> <strong>GetAccessPattern () const</strong></td>
<td>get the preferred access pattern</td>
</tr>
<tr>
<td>void <strong>SetMediaType (const MediaType &amp;t)</strong></td>
<td>set optional media type of stream content</td>
</tr>
<tr>
<td>const MediaType &amp; <strong>GetMediaType () const</strong></td>
<td></td>
</tr>
</tbody>
</table>
get optional media type

virtual bool Open ()
open the stream

virtual void Close ()
close the stream

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

virtual bool Eof () const
return true if end-of-stream reached

virtual void * Map ()
map stream to memory

virtual void Unmap ()
unmap 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

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 <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>
<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
void IO::Stream::SetURI(const URI & u )

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

get stream URI

Get the URI of the stream as string.

```cpp
bool IO::Stream::CanRead() const [virtual]

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.

```cpp
bool IO::Stream::CanWrite() const [virtual]

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

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

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)

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

get the size of the stream in bytes

This method returns the size of the stream in bytes.

Reimplemented in IO::FileStream, IO::MemoryStream, and IO::ZipFileStream.
get the current position of the read/write cursor

This method returns the current position of the read/write cursor.

Reimplemented in \texttt{IO::FileStream}, \texttt{IO::MemoryStream}, and \texttt{IO::ZipFileStream}.

\begin{verbatim}
void
IO::Stream::SetAccessMode( AccessMode m )
\end{verbatim}

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.

\begin{verbatim}
Stream::AccessMode
IO::Stream::GetAccessMode() const
\end{verbatim}

get the access mode of the stream

Get the access mode of the stream.

\begin{verbatim}
void
IO::Stream::SetAccessPattern( AccessPattern p )
\end{verbatim}

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.

\begin{verbatim}
Stream::AccessPattern
IO::Stream::GetAccessPattern() const
\end{verbatim}

get the prefered access pattern

Get the currently set prefered access pattern of the stream.

\begin{verbatim}
bool
IO::Stream::Open() [virtual]
\end{verbatim}
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.

```
void IO::Stream::Close() [virtual]
```

close the stream

Closes the stream.

Reimplemented in IO::FileStream, IO::MemoryStream, and IO::ZipFileStream.

```
bool IO::Stream::IsOpen() const
```

return true if currently open

Return true if the stream is currently open.

```
void IO::Stream::Write(const void *ptr, Size numBytes) [virtual]
```

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(void *ptr, Size numBytes) [virtual]
```
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]
```

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.
IO::StreamReader
IO::StreamReader Class Reference

#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><strong>StreamReader()</strong></td>
<td><strong>constructor</strong></td>
</tr>
<tr>
<td>virtual ~StreamReader()</td>
<td><strong>destructor</strong></td>
</tr>
<tr>
<td>void SetStream (const Ptr&lt; Stream &gt; &amp;s)</td>
<td><strong>set stream to read from</strong></td>
</tr>
<tr>
<td>const Ptr&lt; Stream &gt; &amp; GetStream () const</td>
<td><strong>get currently set stream</strong></td>
</tr>
<tr>
<td>bool HasStream () const</td>
<td><strong>return true if a stream is set</strong></td>
</tr>
<tr>
<td>bool Eof () const</td>
<td><strong>return true if the stream has reached EOF</strong></td>
</tr>
<tr>
<td>virtual bool Open ()</td>
<td><strong>begin reading from the stream</strong></td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td><strong>end reading from the stream</strong></td>
</tr>
<tr>
<td>bool IsOpen () const</td>
<td><strong>return true if currently open</strong></td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td><strong>get the current refcount</strong></td>
</tr>
<tr>
<td>void AddRef ()</td>
<td><strong>increment refcount by one</strong></td>
</tr>
<tr>
<td>void Release ()</td>
<td><strong>decrement refcount and destroy object if refcount is zero</strong></td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td><strong>return true if this object is instance of given class</strong></td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::String &amp;className) const</td>
<td><strong>return true if this object is instance of given class by string</strong></td>
</tr>
<tr>
<td>bool IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td><strong>return true if this object is instance of given class by fourcc</strong></td>
</tr>
<tr>
<td>bool IsA (const Rtti &amp;rtti) const</td>
<td><strong>return true if this object is instance of given class</strong></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>
</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

```c++
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**.

```c++
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.

```c++
bool IO::StreamReader::HasStream() const
```

**return true if a stream is set**

Returns true if a stream is attached to the reader.

```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.
IO::StreamWriter
#include <streamwriter.h>

Inheritance diagram for IO::StreamWriter:
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 <strong>~StreamWriter ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>SetStream (const Ptr&lt; Stream &gt; &amp;s)</strong></td>
<td>set stream to write to</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; Stream &gt; &amp; GetStream () const</strong></td>
<td>get currently set stream</td>
</tr>
<tr>
<td>bool <strong>HasStream () const</strong></td>
<td>return true if a stream is set</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 () const</strong></td>
<td>return true if currently open</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 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>
</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

<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 (NEBU3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
set stream to write to

Attaches the writer to a stream. This will increment the refcount of the stream.

Reimplemented in Messaging::MessageWriter.

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.

return true if a stream is set

Returns true if a stream is attached to the writer.

get the current refcount

Return the current refcount of the object.
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::TextReader
#include <textreader.h>

Inheritance diagram for IO::TextReader:

```
Core::RefCount
  |
  v
IO::StreamReader
  |
  v
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>Type</th>
<th>Member Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td><code>ReadChar()</code></td>
<td>read a single character from the stream</td>
</tr>
<tr>
<td><code>Util::String</code></td>
<td><code>ReadLine()</code></td>
<td>read until next newline</td>
</tr>
<tr>
<td><code>Util::String</code></td>
<td><code>ReadAll()</code></td>
<td>read entire stream into a string object</td>
</tr>
<tr>
<td><code>Util::Array&lt;Util::String&gt;</code></td>
<td><code>ReadAllLines()</code></td>
<td>read entire stream as lines into string array</td>
</tr>
<tr>
<td>void</td>
<td><code>SetStream()</code></td>
<td>set stream to read from</td>
</tr>
<tr>
<td>const <code>Ptr&lt;Stream&gt;</code>   &amp;</td>
<td><code>GetStream()</code></td>
<td>get currently set stream</td>
</tr>
<tr>
<td>bool</td>
<td><code>HasStream()</code></td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td>bool</td>
<td><code>Eof()</code></td>
<td>return true if the stream has reached EOF</td>
</tr>
<tr>
<td>virtual bool</td>
<td><code>Open()</code></td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>Close()</code></td>
<td>end reading from the stream</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsOpen()</code></td>
<td>return true if currently open</td>
</tr>
<tr>
<td>int</td>
<td><code>GetRefCount()</code></td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td><code>AddRef()</code></td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void</td>
<td><code>Release()</code></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td><code>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><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</td>
<td>IsInstanceOf (const Util::FourCC &amp;classFourCC) const</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------------------------</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>bool</td>
<td>IsA (const Rtti &amp;rtti) const</td>
<td></td>
</tr>
<tr>
<td></td>
<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>
<td></td>
</tr>
<tr>
<td></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</td>
<td>IsA (const Util::FourCC &amp;rttiFourCC) const</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>
<td></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetClassName () const</td>
<td></td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () const</td>
<td></td>
</tr>
<tr>
<td></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 DumpRefCountingLeaks()</td>
<td>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
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.

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.

bool IO::StreamReader::HasStream() const [inherited]

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::TextWriter
IO::TextWriter Class Reference

#include <textwriter.h>

Inheritance diagram for IO::TextWriter:

```
Core::RefCounted
   |
   v
IO::StreamWriter
   |
   v
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>void WriteChar(unsigned char c)</td>
<td>write a single character</td>
</tr>
<tr>
<td>void WriteString(const Util::String &amp;str)</td>
<td>write a string</td>
</tr>
<tr>
<td>void WriteFormatted(const char *fmtString,...)</td>
<td>write some formatted text</td>
</tr>
<tr>
<td>void WriteLine(const Util::String &amp;line)</td>
<td>write a line of text and append a new-line</td>
</tr>
<tr>
<td>void WriteLines(const Util::Array<a href="">Util::String</a>&amp; lines)</td>
<td>write a number of lines, separated by new-lines</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; 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>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)</td>
<td>const</td>
</tr>
</tbody>
</table>
### Return if this object is instance of given class by string

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf</strong> (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</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 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><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>

### Get ClassName

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

### Get Class Four CC Code

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

<table>
<thead>
<tr>
<th>Type</th>
<th>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>
set stream to write to

Attaches the writer to a stream. This will increment the refcount of the stream.

Reimplemented in Messaging::MessageWriter.

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.

return true if a stream is set

Returns true if a stream is attached to the writer.

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

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

**IO::URI**
#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
http://www.myserver.com/query?user=bla
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><strong>URI ()</strong></td>
<td>default constructor</td>
</tr>
<tr>
<td><strong>URI (const Util::String &amp;s)</strong></td>
<td>init constructor</td>
</tr>
<tr>
<td>*<em>URI (const char <em>s)</em></em></td>
<td>init constructor</td>
</tr>
<tr>
<td><strong>URI (const URI &amp;rhs)</strong></td>
<td>copy constructor</td>
</tr>
<tr>
<td>void <strong>operator=</strong> (const URI &amp;rhs)</td>
<td>assignment operator</td>
</tr>
<tr>
<td>bool <strong>operator==</strong> (const URI &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool <strong>operator!=</strong> (const URI &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td>void <strong>Set</strong> (const Util::String &amp;s)</td>
<td>set complete URI string</td>
</tr>
<tr>
<td><strong>Util::String AsString ()</strong> const</td>
<td>return as concatenated string</td>
</tr>
<tr>
<td>bool <strong>IsEmpty ()</strong> const</td>
<td>return true if the URI is empty</td>
</tr>
<tr>
<td>bool <strong>IsValid ()</strong> const</td>
<td>return true if the URI is not empty</td>
</tr>
<tr>
<td>void <strong>Clear ()</strong></td>
<td>clear the URI</td>
</tr>
<tr>
<td>void <strong>SetScheme</strong> (const Util::String &amp;s)</td>
<td>set Scheme component (ftp, http, etc...)</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>Scheme ()</strong> const</td>
<td>get Scheme component (default is file)</td>
</tr>
<tr>
<td>void <strong>SetUserInfo</strong> (const Util::String &amp;s)</td>
<td>set UserInfo component</td>
</tr>
<tr>
<td>const Util::String &amp; <strong>UserInfo ()</strong> const</td>
<td>get UserInfo component (can be empty)</td>
</tr>
<tr>
<td>void <strong>SetHost</strong> (const Util::String &amp;s)</td>
<td></td>
</tr>
</tbody>
</table>
const `Util::String` & **Host** () const
get Host component (can be empty)

void **SetPort** (const `Util::String` &s)
set Port component

const `Util::String` & **Port** () const
get Port component (can be empty)

void **SetLocalPath** (const `Util::String` &s)
set LocalPath component

const `Util::String` & **LocalPath** () const
get LocalPath component (can be empty)

void **AppendLocalPath** (const `Util::String` &pathComponent)
append an element to the local path component

void **SetFragment** (const `Util::String` &s)
set Fragment component

const `Util::String` & **Fragment** () const
get Fragment component (can be empty)

void **SetQuery** (const `Util::String` &s)
set Query component

const `Util::String` & **Query** () const
get Query component (can be empty)

`Util::Dictionary < Util::String, Util::String >` **ParseQuery** () const
parse query parameters into a dictionary

`Util::String` **GetTail** () const
get the "tail" (path, query and fragment)
Member Function Documentation

void IO::URI::AppendLocalPath (Util::String pathComponent)

append an element to the local path component

Appends an element to the local path. Automatically inserts a path delimiter "/".

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.

String IO::URI::GetTail ( ) const

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.
IO::XmlReader
IO::XmlReader Class Reference

#include <xmlreader.h>

Inheritance diagram for 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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XmlReader ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~XmlReader ()</strong></td>
<td>destructor</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>HasNode</strong> (const Util::String &amp;path) const</td>
<td>return true if node exists</td>
</tr>
<tr>
<td>Util::String <strong>GetCurrentNodeName</strong> () const</td>
<td>get short name of current node</td>
</tr>
<tr>
<td>Util::String <strong>GetCurrentNodePath</strong> () const</td>
<td>get path to current node</td>
</tr>
<tr>
<td>int <strong>GetCurrentNodeLineNumber</strong> () const</td>
<td>returns the line number of the current node</td>
</tr>
<tr>
<td>void <strong>SetToNode</strong> (const Util::String &amp;path)</td>
<td>set current node as path</td>
</tr>
<tr>
<td>bool <strong>SetToFirstChild</strong> (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 <strong>SetToNextChild</strong> (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 <strong>SetToParent</strong> ()</td>
<td>set current node to parent, return false if no parent exists</td>
</tr>
<tr>
<td>bool <strong>HasAttr</strong> (const Util::String &amp;attr) const</td>
<td>return true if matching attribute exists on current node</td>
</tr>
<tr>
<td>Util::Array <a href="">Util::String</a> <strong>GetAttrs</strong> () const</td>
<td>return names of all attrs on current node</td>
</tr>
<tr>
<td>bool <strong>HasContent</strong> () const</td>
<td>return true if current node has embedded content</td>
</tr>
<tr>
<td>Util::String <strong>GetContent</strong> () const</td>
<td>return embedded content of current node</td>
</tr>
</tbody>
</table>
### Utility Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Util::String</code></td>
<td><code>GetString</code> (const <code>Util::String</code> &amp;attr) const</td>
<td>Get string attribute value from current node</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>GetBool</code> (const <code>Util::String</code> &amp;attr) const</td>
<td>Get bool attribute value from current node</td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>GetInt</code> (const <code>Util::String</code> &amp;attr) const</td>
<td>Get int attribute value from current node</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>GetFloat</code> (const <code>Util::String</code> &amp;attr) const</td>
<td>Get float attribute value from current node</td>
</tr>
<tr>
<td><code>Math::float4</code></td>
<td><code>GetFloat4</code> (const <code>Util::String</code> &amp;attr) const</td>
<td>Get float4 attribute value from current node</td>
</tr>
<tr>
<td><code>Math::matrix44</code></td>
<td><code>GetMatrix44</code> (const <code>Util::String</code> &amp;attr) const</td>
<td>Get matrix44 attribute value from current node</td>
</tr>
<tr>
<td><code>Util::String</code></td>
<td><code>GetOptString</code> (const <code>Util::String</code> &amp;attr, const <code>Util::String</code> &amp;defaultValue) const</td>
<td>Get optional string attribute value from current node</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>GetOptBool</code> (const <code>Util::String</code> &amp;attr, bool defaultValue) const</td>
<td>Get optional bool attribute value from current node</td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>GetOptInt</code> (const <code>Util::String</code> &amp;attr, int defaultValue) const</td>
<td>Get optional int attribute value from current node</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>GetOptFloat</code> (const <code>Util::String</code> &amp;attr, float defaultValue) const</td>
<td>Get optional float attribute value from current node</td>
</tr>
<tr>
<td><code>Math::float4</code></td>
<td><code>GetOptFloat4</code> (const <code>Util::String</code> &amp;attr, const <code>Math::float4</code> &amp;defaultValue) const</td>
<td>Get optional float4 attribute value from current node</td>
</tr>
<tr>
<td><code>Math::matrix44</code></td>
<td><code>GetOptMatrix44</code> (const <code>Util::String</code> &amp;attr, const <code>Math::matrix44</code> &amp;defaultValue) const</td>
<td>Get optional matrix44 attribute value from current node</td>
</tr>
</tbody>
</table>

### Stream Management

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
<td><code>SetStream</code> (const <code>Ptr&lt; Stream &gt;</code> &amp;s)</td>
<td>Set stream to read from</td>
</tr>
<tr>
<td><code>const</code></td>
<td><code>GetStream</code> () const</td>
<td>Get currently set stream</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>HasStream</code> () const</td>
<td>Return true if a stream is set</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>Eof</code> () 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>bool IsOpen () const</td>
<td>return true if currently open</td>
<td></td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
<td></td>
</tr>
<tr>
<td>void AddRef ()</td>
<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>
<td></td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</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>
<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>
<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

static void DumpRefCountingLeaks ()

dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
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 string& attr) 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
string
IO::XmlReader::GetString(const string& attr) 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 existance).

```cpp
bool
IO::XmlReader::GetBool(const string& attr) 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 existance).

```cpp
int
IO::XmlReader::GetInt(const string& attr) 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 existance).

```cpp
float
IO::XmlReader::GetFloat(const string& attr) 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 existance).

```cpp
float4 IO::XmlReader::GetFloat4(const Util::String &attr) 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 existance).

```cpp
matrix44 IO::XmlReader::GetMatrix44(const Util::String &attr) 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 existance).

```cpp
String IO::XmlReader::GetOptString(const Util::String &attr, const Util::String &defaultValue) const
```

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.

```cpp
bool IO::XmlReader::GetOptBool(const Util::String &attr, bool defaultValue) const
```

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.

```cpp
int IO::XmlReader::GetOptInt(const Util::String &attr, int defaultValue const)
```

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.

```cpp
float IO::XmlReader::GetOptFloat(const Util::String &attr, float defaultValue const)
```

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.

```cpp
float4 IO::XmlReader::GetOptFloat4(const Util::String &attr, const Math::float4 &defaultValue &
const)
```

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 Util::String &attr, const Math::matrix44 &defaultValue &
const)
```
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
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::XmlWriter
IO::XmlWriter Class Reference

#include <xmlwriter.h>

Inheritance diagram for IO::XmlWriter:

```
Core::RefCounted
    ▼
IO::StreamWriter
    ▼
IO::XmlWriter
```
Detailed Description

Write XML-formatted data to a stream.

(C) 2006 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XmlWriter ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~XmlWriter ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual bool Open ()</td>
<td>begin writing the stream</td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td>end writing the stream</td>
</tr>
<tr>
<td>bool BeginNode (nodeName)</td>
<td>begin a new node under the current node</td>
</tr>
<tr>
<td>void EndNode ()</td>
<td>end current node, set current node to parent</td>
</tr>
<tr>
<td>void WriteContent (text)</td>
<td>write content text</td>
</tr>
<tr>
<td>void SetString (name, value)</td>
<td>set string attribute on current node</td>
</tr>
<tr>
<td>void SetBool (name, value)</td>
<td>set bool attribute on current node</td>
</tr>
<tr>
<td>void SetInt (name, value)</td>
<td>set int attribute on current node</td>
</tr>
<tr>
<td>void SetFloat (name, value)</td>
<td>set float attribute on current node</td>
</tr>
<tr>
<td>void SetFloat4 (name, value)</td>
<td>set float4 attribute on current node</td>
</tr>
<tr>
<td>void SetMatrix44 (name, value)</td>
<td>set matrix44 attribute on current node</td>
</tr>
<tr>
<td>void SetStream (s)</td>
<td>set stream to write to</td>
</tr>
<tr>
<td>const Stream &amp; GetStream ()</td>
<td>get currently set stream</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool HasStream () const</td>
<td>return true if a stream is set</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 four cc</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 four cc</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>
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.
write content text

Write inline text at current position.

set string attribute on current node

Set the provided attribute to a string value.

set bool attribute on current node

Set the provided attribute to a bool value.

set int attribute on current node

Set the provided attribute to an int value.

set float attribute on current node

Set the provided attribute to a 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.

    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.

```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

IO::ZipArchive
IO::ZipArchive Class Reference

#include <ziparchive.h>

Inheritance diagram for IO::ZipArchive:

```
  Core::RefCounted
     |
     v
IO::ZipArchive
```
Detailed Description

Private helper class for ZipFileSystem to hold per-Zip-archive data. Uses the zlib and the minizip lib for zip file access.

(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>virtual <strong>~ZipArchive ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>SetURI (const URI &amp;n)</strong></td>
<td>set the uri of the zip archive</td>
</tr>
<tr>
<td><strong>GetURI () const</strong></td>
<td>get the uri of the zip archive</td>
</tr>
<tr>
<td><strong>SetPassword (const Util::String &amp;pw)</strong></td>
<td>set optional password</td>
</tr>
<tr>
<td><strong>GetPassword () const</strong></td>
<td>get optional password</td>
</tr>
<tr>
<td><strong>Open ()</strong></td>
<td>open the zip archive</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>close the zip archive</td>
</tr>
<tr>
<td><strong>IsOpen () const</strong></td>
<td>return true if the zip archive is open</td>
</tr>
<tr>
<td><strong>Ptr&lt; ZipFileEntry &gt; FindFileEntry (const Util::String &amp;pathInZipArchive)</strong></td>
<td>find a file entry in the zip archive, return 0 if not exists</td>
</tr>
<tr>
<td><strong>Ptr&lt; ZipDirEntry &gt; FindDirEntry (const Util::String &amp;pathInZipArchive)</strong></td>
<td>find a directory entry in the zip archive, return 0 if not exists</td>
</tr>
<tr>
<td><strong>Util::Array&lt; Util::String &gt; ListFiles (const Util::String &amp;dirPathInZipArchive, const Util::String &amp;pattern)</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;dirPathInZipArchive, const Util::String &amp;pattern)</strong></td>
<td>list all subdirectories in a directory in the archive</td>
</tr>
<tr>
<td><strong>URI ConvertToZipURI (const URI &amp;fileURI) const</strong></td>
<td>convert a “file:” URI into a “zip:” URI pointing into this archive</td>
</tr>
<tr>
<td><strong>ConvertToPathInZipArchive (const Util::String</strong></td>
<td></td>
</tr>
<tr>
<td>Util::String &amp;absPath) const</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
<tr>
<td>convert an absolute path to local path inside zip archive, returns empty string if absPath doesn’t point into this archive</td>
<td></td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td></td>
</tr>
<tr>
<td>get the current refcount</td>
<td></td>
</tr>
<tr>
<td>void AddRef ()</td>
<td></td>
</tr>
<tr>
<td>increment refcount by one</td>
<td></td>
</tr>
<tr>
<td>void Release ()</td>
<td></td>
</tr>
<tr>
<td>decrement refcount and destroy object if refcount is zero</td>
<td></td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &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 IsInstanceOf (const Util::String &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 IsInstanceOf (const Util::FourCC &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 IsA (const Rtti &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 IsA (const Util::String &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 IsA (const Util::FourCC &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 Util::String &amp; GetClassName () const</td>
<td></td>
</tr>
<tr>
<td>get the class name</td>
<td></td>
</tr>
<tr>
<td>Util::FourCC GetClassFourCC () const</td>
<td></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 IO::ZipArchive::Open()

open the zip archive

This opens the zip archive and reads the table of content as a tree of ZipDirEntry and ZipFileEntry objects.
```

```cpp
void IO::ZipArchive::Close()

close the zip archive

This closes the zip archive, releasing the table of contents and closing the zip file.
```

```cpp
URI IO::ZipArchive::ConvertToZipURI(const URI& fileURI) const

convert a "file:" URI into a "zip:" URI pointing into this archive

This method takes a normal "file:" scheme URI and converts 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.
```

```cpp
String IO::ZipArchive::ConvertToPathInZipArchive(const Util::String& absPath) const

convert an absolute path to local path inside zip 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.
```
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.

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:49 2008
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

IO::ZipDirEntry
#include <zipdirentry.h>

Inheritance diagram for IO::ZipDirEntry:

```
Core::RefCounted

O::ZipDirEntry
```
Detailed Description

A directory entry in a zip archive.

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

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ZipDirEntry()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>AddFileEntry(const Util::String &amp;name, const Ptr&lt;ZipFileEntry&gt; &amp;fileEntry)</code></td>
<td>Add a file child entry</td>
</tr>
<tr>
<td><code>AddDirEntry(const Util::String &amp;name, const Ptr&lt;ZipDirEntry&gt; &amp;dirEntry)</code></td>
<td>Add a directory child entry</td>
</tr>
<tr>
<td><code>FindFileEntry(const Util::String &amp;name)</code></td>
<td>Find a direct child file entry, return 0 if not exists</td>
</tr>
<tr>
<td><code>FindDirEntry(const Util::String &amp;name)</code></td>
<td>Find a direct child directory entry, return 0 if not exists</td>
</tr>
<tr>
<td><code>GetDirEntries() const</code></td>
<td>Access to dir entries</td>
</tr>
<tr>
<td><code>GetFileEntries() const</code></td>
<td>Access to file entries</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></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,</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</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 IO::ZipDirEntry::AddFileEntry(const Util::String & name, const Ptr<ZipFileEntry> & fileEntry)

add a file child entry

Adds a new file entry object to the internal dictionary. NOTE: this method will not check whether the entry already exists for performance reasons (doing this would force the dictionary to be sorted after every insert).

void IO::ZipDirEntry::AddDirEntry(const Util::String & name, const Ptr<ZipDirEntry> & dirEntry)

add a directory child entry

Adds a new directory entry object to the internal dictionary. NOTE: this method will not check whether the entry already exists for performance reasons (doing this would force the dictionary to be sorted after every insert).

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.
IO::ZipFileEntry
# include <zipfileentry.h>

Inheritance diagram for IO::ZipFileEntry:
Detailed Description

A file entry in a zip archive.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ZipFileEntry()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~ZipFileEntry()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>IO::Stream::Size GetFileSize()</code></td>
<td>get the uncompressed file size in bytes</td>
</tr>
<tr>
<td><code>bool Read(void *buf, IO::Stream::Size numBytes, const Util::String &amp;password=&quot;&quot;)</code></td>
<td>read the <em>entire</em> content into the provided memory buffer</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</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
bool IO::ZipFileEntry::Read( void * buf,
    IO::Stream::Size numBytes,
    const Util::String & password = ""
)
```

read the *entire* content into the provided memory buffer

NOTE: this method only supports reading the entire file at once, thus numBytes must be equal to the result of `GetFileSize()`!

```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

Increment 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

IO::ZipFileStream
#include <zipfilestream.h>

Inheritance diagram for IO::ZipFileStream:
Detailed Description

Wraps a file in a zip archive into a stream. Allows random access to the file by caching the entire file contents into RAM (the zip filesystem doesn't allow seeking in files). Also note that ZipFileStreams are read-only.

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&pwm=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>Enum Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td><code>AccessMode</code></td>
<td>access modes</td>
</tr>
<tr>
<td>enum</td>
<td><code>AccessPattern</code></td>
<td>access preferred pattern</td>
</tr>
<tr>
<td>enum</td>
<td><code>SeekOrigin</code></td>
<td>seek origins</td>
</tr>
<tr>
<td>typedef int</td>
<td><code>Position</code></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>ZipFileStream()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~ZipFileStream()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual bool <code>CanRead()</code> const</td>
<td>memory streams support reading</td>
</tr>
<tr>
<td>virtual bool <code>CanWrite()</code> const</td>
<td>memory streams support writing</td>
</tr>
<tr>
<td>virtual bool <code>CanSeek()</code> const</td>
<td>memory streams support seeking</td>
</tr>
<tr>
<td>virtual bool <code>CanBeConverted()</code> const</td>
<td>memory streams are mappable</td>
</tr>
<tr>
<td>virtual Size <code>GetSize()</code> const</td>
<td>get the size of the stream in bytes</td>
</tr>
<tr>
<td>virtual <code>GetPosition()</code> const</td>
<td>get the current position of the read/write cursor</td>
</tr>
<tr>
<td>virtual bool <code>Open()</code></td>
<td>open the stream</td>
</tr>
<tr>
<td>virtual void <code>Close()</code></td>
<td>close the stream</td>
</tr>
<tr>
<td>virtual Size <code>Read</code> (void *ptr, Size numBytes)</td>
<td>directly read from the stream</td>
</tr>
<tr>
<td>virtual void <code>Seek</code> (Offset offset, SeekOrigin origin)</td>
<td>seek in stream</td>
</tr>
<tr>
<td>virtual bool <code>Eof()</code> const</td>
<td>return true if end-of-stream reached</td>
</tr>
<tr>
<td>virtual void * <code>Map()</code></td>
<td>map for direct memory-access</td>
</tr>
<tr>
<td>virtual void <code>Unmap</code></td>
<td>unmap a mapped stream</td>
</tr>
<tr>
<td>void <code>SetURI</code> (const URI &amp;u)</td>
<td>set stream location as URI</td>
</tr>
<tr>
<td>const URI &amp; <code>GetURI()</code> const</td>
<td></td>
</tr>
</tbody>
</table>
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)
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 string</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>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
</table>

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

```cpp
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`.

```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 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 prefered 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)
```

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**.

```c++
Stream::Size 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 then numBytes, or 0 if end-of-stream is reached.

```c++
void IO::Stream::Seek (Offset offset, SeekOrigin origin)
```

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**.

```c++
void IO::Stream::Flush ( )
```

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**.

```c++
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 Tue Feb 19 12:16:49 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

**IO::ZipFileSystem**
IO::ZipFileSystem Class Reference

#include <zipfilesystem.h>

Inheritance diagram for IO::ZipFileSystem:

```
Core::RefCounted
  ↓
IO::ZipFileSystem
```
Detailed Description

A filesystem wrapper for ZIP files. It is recommended to use the ZipStreamClass to access Zip files, and only fall back to the ZipFileSystem class if advanced functionality is needed which is not provided by the ZipStreamClass.

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 approach 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><strong>Ptr&lt; ZipArchive &gt; Mount (const URI &amp;uri)</strong></td>
<td>&quot;mount&quot; a zip archive</td>
</tr>
<tr>
<td><strong>void Unmount (const URI &amp;uri)</strong></td>
<td>&quot;unmount&quot; a zip archive by its file name</td>
</tr>
<tr>
<td><strong>void Unmount (const Ptr&lt; ZipArchive &gt; &amp;zipArchive)</strong></td>
<td>&quot;unmount a zip archive by pointer</td>
</tr>
<tr>
<td><strong>bool IsMounted (const URI &amp;uri) const</strong></td>
<td>return true if a zip archive is already mounted</td>
</tr>
<tr>
<td><strong>Util::Array&lt; Ptr &lt; ZipArchive &gt; &gt; GetMountedZipArchives () const</strong></td>
<td>get all mounted zip archives</td>
</tr>
<tr>
<td><strong>const Ptr&lt; ZipArchive &gt; &amp; FindZipArchive (const URI &amp;zipUri)</strong>*</td>
<td>find a zip archive by name, returns 0 if not exists</td>
</tr>
<tr>
<td><strong>const Ptr&lt; ZipArchive &gt; &amp; FindZipArchiveWithFile (const URI &amp;fileUri)</strong></td>
<td>find first zip archive which contains the file path</td>
</tr>
<tr>
<td><strong>const Ptr&lt; ZipArchive &gt; &amp; FindZipArchiveWithDir (const URI &amp;dirUri)</strong></td>
<td>find first zip archive which contains the directory path</td>
</tr>
<tr>
<td><strong>URI ConvertFileToZipURIIfExists (const URI &amp;uri)</strong></td>
<td>transparently convert a <em>URI</em> pointing to a file into a matching zip <em>URI</em></td>
</tr>
<tr>
<td><strong>URI ConvertDirToZipURIIfExists (const URI &amp;uri)</strong></td>
<td>transparently convert a <em>URI</em> pointing to a directory into a matching <em>URI</em></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>bool</td>
<td><code>IsInstanceOf</code> (const <code>Rtti</code> &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><code>IsInstanceOf</code> (const <code>Util::String</code> &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><code>IsInstanceOf</code> (const <code>Util::FourCC</code> &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><code>IsA</code> (const <code>Rtti</code> &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><code>IsA</code> (const <code>Util::String</code> &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><code>IsA</code> (const <code>Util::FourCC</code> &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 <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 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< ZipArchive >
IO::ZipFileSystem::Mount ( const URI & uri )
```

"mount" a zip archive

This "mounts" a zip file of the given filename by creating a `ZipArchive` object and reading the zip file structure. If opening the zip archive fails, 0 will be returned.

```cpp
void
IO::ZipFileSystem::Unmount ( const URI & uri )
```

"unmount" a zip archive by its file name

Unmount a zip archive by zip filename.

```cpp
void
IO::ZipFileSystem::Unmount ( const Ptr< ZipArchive > & zipArchive )
```

"unmount a zip archive by pointer

Unmount a zip archive, this will close the `ZipArchive` object and remove it from the internal registry.

```cpp
const Ptr< ZipArchive > &
IO::ZipFileSystem::FindZipArchive ( const URI & uri )
```

find a zip archive by name, returns 0 if not exists

Resolve a zip archive path into a `ZipArchive` pointer. Returns 0 if no archive with that name exists. The filename will be resolved into an absolute path internally before the lookup happens.

```cpp
const Ptr< ZipArchive > &
IO::ZipFileSystem::FindZipArchiveWithFile ( const URI & uri )
```
find first zip 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 the name of 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
const Ptr< ZipArchive > & IO::ZipFileSystem::FindZipArchiveWithFile( const URI & uri )
```

find first zip archive which contains the directory path

Same as **FindZipArchiveWithFile()**, but checks for a directory entry in a zip file.

```cpp
URI IO::ZipFileSystem::ConvertFileToZipURIIfExists( const URI & uri )
```

transparently convert a **URI** pointing to a file into a matching zip **URI**

This checks if the provided file **URI** points into one of the mounted zip archives, and if yes returns a matching zip **URI** into the archive. If no match exists, the original **URI** will be returned.

```cpp
URI IO::ZipFileSystem::ConvertDirToZipURIIfExists( const URI & uri )
```

transparently convert a **URI** pointing to a directory into a matching zip **URI**

This checks if the provided directory **URI** points into one of the mounted zip archives, and if yes returns a matching zip **URI** into the archive. If no match exists, the original **URI** will be returned.

```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.
Legacy::Nax2StreamReader
Legacy::Nax2StreamReader Class Reference

#include <nax2streamreader.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
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::AbstractLightEntity
Lighting::AbstractLightEntity Class Reference

#include <abstractlightentity.h>

Inheritance diagram for Lighting::AbstractLightEntity:

[Diagram showing the inheritance hierarchy with classes Core::RefCounted, Graphics::GraphicsEntity, Lighting::AbstractLightEntity, Lighting::GlobalLightEntity, and Lighting::SpotLightEntity connected by arrows.]
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>Enum</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Type</strong></td>
<td>Entity types</td>
</tr>
<tr>
<td></td>
<td><strong>LinkType</strong></td>
<td>Visibility link types</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AbstractLightEntity ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~AbstractLightEntity ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>LightType::Code GetLightType () const</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 () const</code></td>
<td>Get primary light color</td>
</tr>
<tr>
<td><code>void SetProjMapUvOffsetAndScale (const Math::float4 &amp;v)</code></td>
<td>Set projection map UV offset and scale (xy-offset, zw-scale)</td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetProjMapUvOffsetAndScale () const</code></td>
<td>Get projection map UV offset and scale</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>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 () const</code></td>
<td>Get world-to-light-projection transform (transform 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>const Math::float4 &amp; GetShadowBufferUvOffsetAndScale () const</code></td>
<td>Get shadow buffer uv rectangle</td>
</tr>
<tr>
<td><code>bool IsActive () const</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>return true if entity is currently active (is between OnActivate()/OnDeactivate())</code></td>
<td></td>
</tr>
<tr>
<td><code>bool IsValid () const</code></td>
<td>return true if entity is current valid (ready for rendering)</td>
</tr>
<tr>
<td><code>Type</code></td>
<td></td>
</tr>
<tr>
<td><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;</code></td>
<td></td>
</tr>
<tr>
<td><code>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;</code></td>
<td></td>
</tr>
<tr>
<td><code>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>const Ptr&lt; Cell &gt; &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>GetCell () const</code></td>
<td>get the cell this entity is attached to</td>
</tr>
<tr>
<td><code>bool IsAttachedToCell () const</code></td>
<td>return true if entity is attached to cell</td>
</tr>
<tr>
<td><code>const Math::bbox &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>GetLocalBoundingBox () const</code></td>
<td>get the local space bounding box</td>
</tr>
<tr>
<td><code>const Math::bbox &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>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; GraphicsEntity &gt; &amp;entity)</code></td>
<td>add visibility link</td>
</tr>
<tr>
<td><code>const Util::Array &lt; Ptr &lt; GraphicsEntity &gt; &gt; &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>GetLinks (LinkType type) const</code></td>
<td>get visibility links by type</td>
</tr>
<tr>
<td><code>virtual Math::ClipStatus::Type</code></td>
<td></td>
</tr>
<tr>
<td><code>ComputeClipStatus (const Math::bbox &amp;box)</code></td>
<td>compute clip status against bounding box</td>
</tr>
<tr>
<td>Void</td>
<td><code>SetTime(Timing::Time t)</code></td>
</tr>
<tr>
<td>------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Time</td>
<td><code>GetTime()</code> const</td>
</tr>
<tr>
<td>Int</td>
<td><code>GetRefCount()</code> const</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>Util::String &amp;</td>
<td><code>GetClassName()</code> const</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><code>GetClassFourCC()</code> 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

<table>
<thead>
<tr>
<th>virtual void</th>
<th><strong>OnTransformChanged</strong> (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetLightType</strong> (LightType::Code c)</td>
</tr>
<tr>
<td></td>
<td>set the light type (must be called from sub-classes constructor</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetType</strong> (Type t)</td>
</tr>
<tr>
<td></td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetLocalBoundingBox</strong> (const Math::bbox &amp;b)</td>
</tr>
<tr>
<td></td>
<td>set the local space bounding box</td>
</tr>
<tr>
<td>void</td>
<td><strong>SetValid</strong> (bool b)</td>
</tr>
<tr>
<td></td>
<td>set to valid state (when the entity becomes ready for rendering)</td>
</tr>
<tr>
<td>virtual <strong>Ptr</strong> &lt; GraphicsEntity &gt;</td>
<td><strong>CreateClone</strong> (void) const</td>
</tr>
<tr>
<td></td>
<td>create a clone of the entity</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnActivate</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>called when entity is created</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnDeactivate</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>called before entity is destroyed</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnAttachToStage</strong> (const <strong>Ptr</strong>&lt; Stage &gt; &amp;stage)</td>
</tr>
<tr>
<td></td>
<td>called when attached to Stage</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnRemoveFromStage</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>called when removed from Stage</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnAttachToCell</strong> (const <strong>Ptr</strong>&lt; Cell &gt; &amp;cell)</td>
</tr>
<tr>
<td></td>
<td>called when attached to a Cell</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnRemoveFromCell</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>called when removed from a Cell</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnShow</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>called when the entity becomes visible</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnHide</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>called when the entity becomes invisible</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnUpdate</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>called to update the entity before rendering</td>
</tr>
<tr>
<td>virtual void</td>
<td><strong>OnRender</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>called before the entity is rendered</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>OnRenderDebug ()</code></td>
<td>called to render a debug visualization of the entity</td>
</tr>
<tr>
<td><code>void UpdatePositionInCellTree ()</code></td>
<td>update our position in the cell hierarchy</td>
</tr>
<tr>
<td><code>void UpdateGlobalBoundingBox ()</code></td>
<td>update the global bounding box from the transform and local box</td>
</tr>
</tbody>
</table>
Member Function Documentation

**ClipStatus::Type**

*Graphics::GraphicsEntity::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 **Graphics::CameraEntity**, **Lighting::GlobalLightEntity**, and **Lighting::SpotLightEntity**.

**Ptr< GraphicsEntity >**

*Graphics::GraphicsEntity::CreateClone*( ) const [protected, virtual, inherited]

create a clone of the entity

This method is called to create an exact clone of this graphics entity. The new entity will not be attached to a stage.

**void**

*Graphics::GraphicsEntity::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 **Graphics::ActorEntity**, and **Graphics::ModelEntity**.

**void**

*Graphics::GraphicsEntity::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 `Graphics::ActorEntity`, `Graphics::CameraEntity`, and `Graphics::ModelEntity`.

```cpp
void Graphics::GraphicsEntity::OnAttachToStage(const Ptr<Stage> & 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 Graphics::GraphicsEntity::OnRemoveFromStage() [protected, virtual, inherited]
```
called when removed from **Stage**

This method is called when the graphics entity is removed from a stage.

```cpp
void Graphics::GraphicsEntity::OnAttachToCell(const Ptr<Cell> & c) [protected, virtual, inherited]
```
called when attached to a **Cell**

This method is called when the graphics entity is attached to a cell inside a stage. When entity travel through the graphics world, they will be remove and attached themselves from and to Cells as they cross **Cell** borders.

```cpp
void Graphics::GraphicsEntity::OnRemoveFromCell() [protected, virtual, inherited]
```
called when removed from a **Cell**

Called when the graphics entity is removed from a cell inside a stage.

```c++
void Graphics::GraphicsEntity::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.

```c++
void Graphics::GraphicsEntity::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.

```c++
void Graphics::GraphicsEntity::OnUpdate() [protected, virtual, inherited]
```

called to update the entity before rendering

This method is called on the graphics entity to update itself, for instance to implement hierarchy animation or particle updates.

Reimplemented in **Graphics::ActorEntity**, and **Graphics::ModelEntity**.

```c++
void Graphics::GraphicsEntity::OnRender() [protected, virtual, inherited]
```

called before the entity is rendered

This method is called on the graphics entity to render itself. This method will only be called if the entity is visible through the camera of the currently rendered **View**.

```c++
void Graphics::GraphicsEntity::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 `Graphics::ModelEntity`.

```cpp
void Graphics::GraphicsEntity::UpdatePositionInCellTree() [protected, inherited]
```

update our position in the cell hierarchy

This method is called from `OnUpdate()` when either the entity's transformation or bounding box has changed. It checks if the entity still fits into its current cell, and if not, moves the entity in a new cell.

```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::GlobalLightEntity
Lighting::GlobalLightEntity Class Reference

#include <globallightentity.h>

Inheritance diagram for Lighting::GlobalLightEntity:

```
Core::RefCounted

Graphics::GraphicsEntity

Lighting::AbstractLightEntity

Lighting::GlobalLightEntity
```
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>enum</th>
<th>Type</th>
<th>entity types</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>LinkType</td>
<td>visibility link types</td>
</tr>
</tbody>
</table>
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GlobalLightEntity ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual Math::ClipStatus::Type</strong></td>
<td>ComputeClipStatus (const Math::bbox &amp;box) compute clip status against bounding box</td>
</tr>
<tr>
<td><strong>void SetBackLightColor (const Math::float4 &amp;c)</strong></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>LightType::Code <strong>GetLightType ()</strong> const</td>
<td>get the light type</td>
</tr>
<tr>
<td><strong>void SetColor (const Math::float4 &amp;c)</strong></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><strong>void 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 ()</strong> const</td>
<td>get projection map UV offset and scale</td>
</tr>
<tr>
<td><strong>void SetCastShadows (bool b)</strong></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>const Math::matrix44 &amp; <strong>GetInvTransform ()</strong> const</td>
<td>get inverse transform (transforms from world to light space)</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; <strong>GetProjTransform ()</strong> const</td>
<td>get light-projection matrix (transforms from light space to light projection space)</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; <strong>GetInvLightProjTransform ()</strong> const</td>
<td>get world-to-light-projection transform (transform from world to light projection space)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>void <code>SetShadowBufferUvOffsetAndScale(const Math::float4 &amp;uvOffset)</code></td>
<td>set shadow buffer uv rectangle (optionally set by light/shadow servers)</td>
</tr>
<tr>
<td>const Math::float4 &amp; <code>GetShadowBufferUvOffsetAndScale()</code> const</td>
<td>get shadow buffer uv rectangle</td>
</tr>
<tr>
<td>bool <code>IsActive()</code> const</td>
<td>return true if entity is currently active (is between <code>OnActivate()</code>/<code>OnDeactivate()</code>)</td>
</tr>
<tr>
<td>bool <code>IsValid()</code> const</td>
<td>return true if entity is current valid (ready for rendering)</td>
</tr>
<tr>
<td>Type <code>GetType()</code> const</td>
<td>get the entity type</td>
</tr>
<tr>
<td>void <code>SetTransform(const Math::matrix44 &amp;m)</code></td>
<td>set the entity's world space transform</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; <code>GetTransform()</code> const</td>
<td>get the entity's world space transform</td>
</tr>
<tr>
<td>void <code>SetVisible(bool b)</code></td>
<td>set the entity's visibility</td>
</tr>
<tr>
<td>bool <code>IsVisible()</code> const</td>
<td>return true if entity is set to visible</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Stage &gt; &amp;</code> <code>GetStage()</code> const</td>
<td>get the stage this entity is attached to</td>
</tr>
<tr>
<td>bool <code>IsAttachedToStage()</code> const</td>
<td>return true if entity is attached to stage</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Cell &gt; &amp;</code> <code>GetCell()</code> const</td>
<td>get the cell this entity is attached to</td>
</tr>
<tr>
<td>bool <code>IsAttachedToCell()</code> const</td>
<td>return true if entity is attached to cell</td>
</tr>
<tr>
<td>const Math::bbox &amp; <code>GetLocalBoundingBox()</code> const</td>
<td>get the local space bounding box</td>
</tr>
<tr>
<td>const Math::bbox &amp; <code>GetGlobalBoundingBox()</code></td>
<td>get bounding box in global space</td>
</tr>
<tr>
<td>void <code>ClearLinks(LinkType linkType)</code></td>
<td>clear all visibility links</td>
</tr>
<tr>
<td>void <code>AddLink(LinkType linkType, const </code>Ptr&lt; GraphicsEntity &gt; &amp;`)</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>add visibility link</code></td>
<td></td>
</tr>
<tr>
<td><code>const Util::Array&lt;Ptr&lt;GraphicsEntity&gt;&gt; &amp; GetLinks (LinkType type) const</code></td>
<td>get visibility links by type</td>
</tr>
<tr>
<td><code>void SetTime (Timing::Time t)</code></td>
<td>set graphics time</td>
</tr>
<tr>
<td><code>Timing::Time GetTime () const</code></td>
<td>get graphics time</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</th>
<th>DumpRefCountingLeaks ()</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>virtual void OnTransformChanged()</code></td>
<td>called when transform matrix changed</td>
</tr>
<tr>
<td><code>void SetLightType (LightType::Code c)</code></td>
<td>set the light type (must be called from sub-classes constructor)</td>
</tr>
<tr>
<td><code>void SetType (Type t)</code></td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td><code>void SetLocalBoundingBox (const Math::bbox &amp;b)</code></td>
<td>set the local space bounding box</td>
</tr>
<tr>
<td><code>void SetValid (bool b)</code></td>
<td>set to valid state (when the entity becomes ready for rendering)</td>
</tr>
<tr>
<td><code>virtual Ptr &lt; GraphicsEntity &gt; CreateClone () const</code></td>
<td>create a clone of the entity</td>
</tr>
<tr>
<td><code>virtual void OnActivate ()</code></td>
<td>called when entity is created</td>
</tr>
<tr>
<td><code>virtual void OnDeactivate ()</code></td>
<td>called before entity is destroyed</td>
</tr>
<tr>
<td><code>virtual void OnAttachToStage (const Ptr&lt; Stage &gt; &amp;stage)</code></td>
<td>called when attached to Stage</td>
</tr>
<tr>
<td><code>virtual void OnRemoveFromStage ()</code></td>
<td>called when removed from Stage</td>
</tr>
<tr>
<td><code>virtual void OnAttachToCell (const Ptr&lt; Cell &gt; &amp;cell)</code></td>
<td>called when attached to a Cell</td>
</tr>
<tr>
<td><code>virtual void OnRemoveFromCell ()</code></td>
<td>called when removed from a Cell</td>
</tr>
<tr>
<td><code>virtual void OnShow ()</code></td>
<td>called when the entity becomes visible</td>
</tr>
<tr>
<td><code>virtual void OnHide ()</code></td>
<td>called when the entity becomes invisible</td>
</tr>
<tr>
<td><code>virtual void OnUpdate ()</code></td>
<td>called to update the entity before rendering</td>
</tr>
<tr>
<td><code>virtual void OnRender ()</code></td>
<td>called before the entity is rendered</td>
</tr>
<tr>
<td><code>virtual void</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>OnRenderDebug()</code></td>
<td>called to render a debug visualization of the entity</td>
</tr>
<tr>
<td><code>void UpdatePositionInCellTree()</code></td>
<td>update our position in the cell hierarchy</td>
</tr>
<tr>
<td><code>void UpdateGlobalBoundingBox()</code></td>
<td>update the global bounding box from the transform and local box</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Ptr< GraphicsEntity >**

Graphics::GraphicsEntity::CreateClone ( ) const [protected, virtual, inherited]

create a clone of the entity

This method is called to create an exact clone of this graphics entity. The new entity will not be attached to a stage.

**void**

Graphics::GraphicsEntity::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 **Graphics::ActorEntity**, and **Graphics::ModelEntity**.

**void**

Graphics::GraphicsEntity::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 **Graphics::ActorEntity**, **Graphics::CameraEntity**, and **Graphics::ModelEntity**.

**void**

Graphics::GraphicsEntity::OnAttachToStage ( const Ptr< Stage > & ) [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 Graphics::GraphicsEntity::OnRemoveFromStage() [protected, virtual, inherited]
```
called when removed from **Stage**

This method is called when the graphics entity is removed from a stage.

```cpp
void Graphics::GraphicsEntity::OnAttachToCell(const Ptr<Cell>& c) [protected, virtual, inherited]
```
called when attached to a **Cell**

This method is called when the graphics entity is attached to a cell inside a stage. When entity travel through the graphics world, they will be remove and attached themselves from and to Cells as they cross **Cell** borders.

```cpp
void Graphics::GraphicsEntity::OnRemoveFromCell() [protected, virtual, inherited]
```
called when removed from a **Cell**

Called when the graphics entity is removed from a cell inside a stage.

```cpp
void Graphics::GraphicsEntity::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.
Graphics::GraphicsEntity::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.

void Graphics::GraphicsEntity::OnUpdate() [protected, virtual, inherited]

called to update the entity before rendering

This method is called on the graphics entity to update itself, for instance to implement hierarchy animation or particle updates.

Reimplemented in Graphics::ActorEntity, and Graphics::ModelEntity.

void Graphics::GraphicsEntity::OnRender() [protected, virtual, inherited]

called before the entity is rendered

This method is called on the graphics entity to render itself. This method will only be called if the entity is visible through the camera of the currently rendered View.

void Graphics::GraphicsEntity::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 Graphics::ModelEntity.

void Graphics::GraphicsEntity::UpdatePositionInCellTree() [protected, inherited]

update our position in the cell hierarchy
This method is called from `OnUpdate()` when either the entity's transformation or bounding box has changed. It checks if the entity still fits into its current cell, and if not, moves the entity in a new cell.

```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.
Lighting::LightServer
Lighting::LightServer Class Reference

#include <lightserver.h>

Inheritance diagram for Lighting::LightServer:

```
Core::RefCounted

Lighting::LightServerBase

Lighting::SM30LightServer

Lighting::LightServer
```
Detailed Description

The light server collects all lights contributing to the scene and controls the realtime lighting process.

(C) 2007 Radon Labs GmbH
LightServer ()
constructor

virtual ~LightServer ()
destructor

void Open ()
one open the light server

void Close ()
close the light server

void ApplyModelEntityLights (const Ptr<Graphics::ModelEntity> &modelEntity)
apply lighting parameters for a visible model entity

bool IsOpen () const
return true if light server is open

void BeginFrame (const Ptr<Graphics::CameraEntity> &cameraEntity)
begin lighting frame

void BeginAttachVisibleLights ()
begin attaching visible light sources

void AttachVisibleLight (const Ptr<AbstractLightEntity> &lightEntity)
attach a visible light source

void EndAttachVisibleLights ()
end attaching visible light sources

void EndFrame ()
end lighting 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
<table>
<thead>
<tr>
<th>bool</th>
<th><code>IsInstanceOf</code> (const <code>Util::String</code> &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><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</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 Lighting::SM30LightServer::ApplyModelEntityLights(
const Ptr<Graphics::ModelEntity> modelEntity &)
[inherited]

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.

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

getcherurrent 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.

dump 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><strong>LightServerBase ()</strong></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>void BeginFrame (const Ptr&lt; Graphics::CameraEntity &gt; &amp;cameraEntity)</td>
<td>begin lighting frame</td>
</tr>
<tr>
<td>void BeginAttachVisibleLights ()</td>
<td>begin attaching visible light sources</td>
</tr>
<tr>
<td>void AttachVisibleLight (const Ptr&lt; AbstractLightEntity &gt; &amp;lightEntity)</td>
<td>attach a visible light source</td>
</tr>
<tr>
<td>void EndAttachVisibleLights ()</td>
<td>end attaching visible light sources</td>
</tr>
<tr>
<td>void ApplyModelEntityLights (const Ptr&lt; Graphics::ModelEntity &gt; &amp;modelEntity)</td>
<td>apply lighting parameters for a visible model entity</td>
</tr>
<tr>
<td>void EndFrame ()</td>
<td>end lighting frame</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>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 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>
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`.

```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 ( ) 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::LightType
Lighting::LightType Class Reference

#include <lighttype.h>
Detailed Description

Identifies different light types.

(C) 2007 Radon Labs GmbH
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 (const Ptr&lt; Graphics::CameraEntity &gt; &amp;camera)</strong></td>
<td>set camera entity which defines the view and projection transform</td>
</tr>
<tr>
<td><strong>GetCameraEntity () const</strong></td>
<td>get camera entity</td>
</tr>
<tr>
<td><strong>SetLightDir (const Math::vector &amp;dir)</strong></td>
<td>set light direction</td>
</tr>
<tr>
<td><strong>GetLightDir () const</strong></td>
<td>get light direction</td>
</tr>
<tr>
<td><strong>Compute ()</strong></td>
<td>compute PSSM split volumes</td>
</tr>
<tr>
<td><strong>GetSplitLightTransform (IndexT splitIndex) const</strong></td>
<td>get view matrix for a view frustum split (valid after Compute)</td>
</tr>
<tr>
<td><strong>GetSplitProjTransform (IndexT splitIndex) const</strong></td>
<td>get projection transform for a view frustum split (valid after Compute)</td>
</tr>
<tr>
<td><strong>GetSplitLightProjTransform (IndexT splitIndex) const</strong></td>
<td>get light projection transform for given frustum split (valid after Compute)</td>
</tr>
<tr>
<td><strong>GetSplitDistances () const</strong></td>
<td>get raw pointer to split distances</td>
</tr>
<tr>
<td><strong>GetSplitLightProjTransforms () const</strong></td>
<td>get raw pointer to LightProjTransforms</td>
</tr>
</tbody>
</table>
### Static Public Attributes

<table>
<thead>
<tr>
<th>static const SizeT</th>
<th><strong>NumSplits</strong> = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>number of view volume splits</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:49 2008
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><code>ShadowServer ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~ShadowServer ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Open ()</code></td>
<td>Open the shadow server</td>
</tr>
<tr>
<td><code>Close ()</code></td>
<td>Close the shadow server</td>
</tr>
<tr>
<td><code>UpdateShadowBuffers ()</code></td>
<td>Update shadow buffer</td>
</tr>
<tr>
<td><code>GetLocalLightShadowBufferTexture ()</code></td>
<td>Get pointer to shadow buffer for local lights</td>
</tr>
<tr>
<td><code>GetGlobalLightShadowBufferTexture ()</code></td>
<td>Get pointer to PSSM shadow buffer for global lights</td>
</tr>
<tr>
<td><code>GetPSSMSplitDistances ()</code></td>
<td>Get array of PSSM split distances</td>
</tr>
<tr>
<td><code>GetPSSMSplitLightProjTransforms ()</code></td>
<td>Get array of PSSM LightProjTransforms</td>
</tr>
</tbody>
</table>
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.
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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SM30LightServer</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~SM30LightServer</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>Open</strong> ()</td>
<td>open the light server</td>
</tr>
<tr>
<td>void <strong>Close</strong> ()</td>
<td>close the light server</td>
</tr>
<tr>
<td>void <strong>ApplyModelEntityLights</strong> (const Ptr&lt; Graphics::ModelEntity &gt; &amp;modelEntity)</td>
<td>apply lighting parameters for a visible model entity</td>
</tr>
<tr>
<td>bool <strong>IsOpen</strong> () const</td>
<td>return true if light server is open</td>
</tr>
<tr>
<td>void <strong>BeginFrame</strong> (const Ptr&lt; Graphics::CameraEntity &gt; &amp;cameraEntity)</td>
<td>begin lighting frame</td>
</tr>
<tr>
<td>void <strong>BeginAttachVisibleLights</strong> ()</td>
<td>begin attaching visible light sources</td>
</tr>
<tr>
<td>void <strong>AttachVisibleLight</strong> (const Ptr&lt; AbstractLightEntity &gt; &amp;lightEntity)</td>
<td>attach a visible light source</td>
</tr>
<tr>
<td>void <strong>EndAttachVisibleLights</strong> ()</td>
<td>end attaching visible light sources</td>
</tr>
<tr>
<td>void <strong>EndFrame</strong> ()</td>
<td>end lighting frame</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>IsInstanceOf (const Util::String &amp;className) 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>IsInstanceOf (const Util::FourCC &amp;classFourCC) 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>IsA (const Rtti &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>IsA (const Util::String &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>IsA (const Util::FourCC &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 Util::String &amp;</td>
<td>GetClassName () const</td>
</tr>
<tr>
<td></td>
<td><em>get the class name</em></td>
</tr>
<tr>
<td>Util::FourCC</td>
<td>GetClassFourCC () 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
void
Lighting::SM30LightServer::ApplyModelEntityLights ( const Ptr< Graphics::ModelEntity > 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
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
void Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]
```

dump 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 Tue Feb 19 12:16:49 2008
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 &lt; CoreGraphics::Texture &gt; &amp; GetLocalLightShadowBufferTexture ()</strong></td>
<td>Get pointer to shadow buffer for local lights</td>
</tr>
<tr>
<td><strong>const Ptr &lt; CoreGraphics::Texture &gt; &amp; GetGlobalLightShadowBufferTexture ()</strong></td>
<td>Get pointer to PSSM shadow buffer for global lights</td>
</tr>
<tr>
<td><strong>const float * GetPSSMSplitDistances ()</strong></td>
<td>Get array of PSSM split distances</td>
</tr>
<tr>
<td><strong>const Math::matrix44 * GetPSSMSplitLightProjTransforms ()</strong></td>
<td>Get array of PSSM LightProjTransforms</td>
</tr>
</tbody>
</table>
Member Function Documentation

void 
Lighting::SM30ShadowServer::UpdateShadowBuffers();

update shadow buffer

This method updates the internal shadow buffer render targets.

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.
Lighting::SpotLightEntity
Lighting::SpotLightEntity Class Reference

#include <spotlightentity.h>

Inheritance diagram for Lighting::SpotLightEntity:
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>Type</th>
<th>entity types</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>LinkType</td>
<td>visibility link types</td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>SpotLightEntity()</code></td>
<td>Constructor</td>
<td></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>
<td></td>
</tr>
<tr>
<td><code>LightType::Code GetLightType() const</code></td>
<td>Get the light type</td>
<td></td>
</tr>
<tr>
<td><code>void SetColor(const Math::float4 &amp;c)</code></td>
<td>Set primary light color</td>
<td></td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetColor() const</code></td>
<td>Get primary light color</td>
<td></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>
<td></td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetProjMapUvOffsetAndScale() const</code></td>
<td>Get projection map UV offset and scale</td>
<td></td>
</tr>
<tr>
<td><code>void SetCastShadows(bool b)</code></td>
<td>Enable/disable shadow casting</td>
<td></td>
</tr>
<tr>
<td><code>bool GetCastShadows() const</code></td>
<td>Get shadow casting flag</td>
<td></td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetInvTransform() const</code></td>
<td>Get inverse transform (transforms from world to light space)</td>
<td></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>
<td></td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetInvLightProjTransform() const</code></td>
<td>Get world-to-light-projection transform (transform from world to light projection space)</td>
<td></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>
<td></td>
</tr>
<tr>
<td><code>const Math::float4 &amp; GetShadowBufferUvOffsetAndScale() const</code></td>
<td>Get shadow buffer uv rectangle</td>
<td></td>
</tr>
<tr>
<td><code>bool IsActive() const</code></td>
<td>Get shadow buffer uv rectangle</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>IsValid()</code> const</td>
<td>return true if entity is currently valid (ready for rendering)</td>
<td></td>
</tr>
<tr>
<td><code>GetType()</code> const</td>
<td>get the entity type</td>
<td></td>
</tr>
<tr>
<td><code>SetTransform()</code> const</td>
<td>set the entity's world space transform</td>
<td></td>
</tr>
<tr>
<td><code>GetTransform()</code> const</td>
<td>get the entity's world space transform</td>
<td></td>
</tr>
<tr>
<td><code>SetVisible()</code> bool b</td>
<td>set the entity's visibility</td>
<td></td>
</tr>
<tr>
<td><code>IsVisible()</code> const</td>
<td>return true if entity is set to visible</td>
<td></td>
</tr>
<tr>
<td><code>GetStage()</code> const</td>
<td>get the stage this entity is attached to</td>
<td></td>
</tr>
<tr>
<td><code>IsAttachedToStage()</code> const</td>
<td>return true if entity is attached to stage</td>
<td></td>
</tr>
<tr>
<td><code>GetCell()</code> const</td>
<td>get the cell this entity is attached to</td>
<td></td>
</tr>
<tr>
<td><code>IsAttachedToCell()</code> const</td>
<td>return true if entity is attached to cell</td>
<td></td>
</tr>
<tr>
<td><code>GetLocalBoundingBox()</code> const</td>
<td>get the local space bounding box</td>
<td></td>
</tr>
<tr>
<td><code>GetGlobalBoundingBox()</code></td>
<td>get bounding box in global space</td>
<td></td>
</tr>
<tr>
<td><code>ClearLinks()</code> LinkType</td>
<td>clear all visibility links</td>
<td></td>
</tr>
<tr>
<td><code>AddLink()</code> LinkType, <code>Ptr&lt;GraphicsEntity&gt;</code> &amp;entity</td>
<td>add visibility link</td>
<td></td>
</tr>
<tr>
<td><code>GetLinks()</code> LinkType type</td>
<td>get visibility links by type</td>
<td></td>
</tr>
<tr>
<td><code>SetTime()</code> Timing::Time t</td>
<td>set graphics time</td>
<td></td>
</tr>
<tr>
<td><code>GetTime()</code> const</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>int GetRefCount () const</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) const</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) 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>
<tr>
<td>static void DumpRefCountingLeaks ()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></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 <code>OnTransformChanged()</code></td>
<td>called when transform matrix changed</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>SetType</code> (Type t)</td>
<td>set entity type, call in constructor of derived class!</td>
</tr>
<tr>
<td>void <code>SetLocalBoundingBox</code> (const Math::bbox &amp;b)</td>
<td>set the local space bounding box</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>virtual <code>Ptr &lt; GraphicsEntity &gt; CreateClone()</code></td>
<td>create a clone of the entity</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 <code>Ptr &lt; Stage &gt;</code> &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>OnAttachToCell</code> (const <code>Ptr &lt; Cell &gt;</code> &amp;cell)</td>
<td>called when attached to a Cell</td>
</tr>
<tr>
<td>virtual void <code>OnRemoveFromCell</code> ()</td>
<td>called when removed from a Cell</td>
</tr>
<tr>
<td>virtual void <code>OnShow</code> ()</td>
<td>called when the entity becomes visible</td>
</tr>
<tr>
<td>virtual void <code>OnHide</code> ()</td>
<td>called when the entity becomes invisible</td>
</tr>
<tr>
<td>virtual void <code>OnUpdate</code> ()</td>
<td>called to update the entity before rendering</td>
</tr>
<tr>
<td>virtual void <code>OnRender</code> ()</td>
<td>called before the entity is rendered</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OnRenderDebug ()</td>
<td>called to render a debug visualization of the entity</td>
</tr>
<tr>
<td>void UpdatePositionInCellTree ()</td>
<td>update our position in the cell hierarchy</td>
</tr>
<tr>
<td>void UpdateGlobalBoundingBox ()</td>
<td>update the global bounding box from the transform and local box</td>
</tr>
</tbody>
</table>
Member Function Documentation

**Ptr< GraphicsEntity >**
Graphics::GraphicsEntity::CreateClone ( ) const [protected, virtual, inherited]

create a clone of the entity

This method is called to create an exact clone of this graphics entity. The new entity will not be attached to a stage.

**void**
Graphics::GraphicsEntity::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 **Graphics::ActorEntity**, and **Graphics::ModelEntity**.

**void**
Graphics::GraphicsEntity::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 **Graphics::ActorEntity**, **Graphics::CameraEntity**, and **Graphics::ModelEntity**.

**void**
Graphics::GraphicsEntity::OnAttachToStage ( const Ptr< Stage > & 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 Graphics::GraphicsEntity::OnRemoveFromStage() [protected, virtual, inherited]
called when removed from Stage
```

This method is called when the graphics entity is removed from a stage.

```cpp
void Graphics::GraphicsEntity::OnAttachToCell(const Ptr<Cell> & c) [protected, virtual, inherited]
called when attached to a Cell
```

This method is called when the graphics entity is attached to a cell inside a stage. When entity travel through the graphics world, they will be remove and attached themselves from and to Cells as they cross Cell borders.

```cpp
void Graphics::GraphicsEntity::OnRemoveFromCell() [protected, virtual, inherited]
called when removed from a Cell
```

Called when the graphics entity is removed from a cell inside a stage.

```cpp
void Graphics::GraphicsEntity::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.
Graphics::GraphicsEntity::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.

void Graphics::GraphicsEntity::OnUpdate() [protected, virtual, inherited]

called to update the entity before rendering

This method is called on the graphics entity to update itself, for instance to implement hierarchy animation or particle updates.

Reimplemented in Graphics::ActorEntity, and Graphics::ModelEntity.

void Graphics::GraphicsEntity::OnRender() [protected, virtual, inherited]

called before the entity is rendered

This method is called on the graphics entity to render itself. This method will only be called if the entity is visible through the camera of the currently rendered View.

void Graphics::GraphicsEntity::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 Graphics::ModelEntity.

void Graphics::GraphicsEntity::UpdatePositionInCellTree() [protected, inherited]

update our position in the cell hierarchy
This method is called from `OnUpdate()` when either the entity's transformation or bounding box has changed. It checks if the entity still fits into its current cell, and if not, moves the entity in a new cell.

```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

Math::bbox
Math::bbox Class Reference

#include <bbox.h>
Detailed Description

Nebula's bounding box class.

Todo:
   : UNTESTED!

(C) 2004 RadonLabs GmbH
<table>
<thead>
<tr>
<th>enum</th>
<th>clip codes</th>
</tr>
</thead>
</table>

Public Types
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bbox()</strong></td>
<td>Constructor 1</td>
</tr>
<tr>
<td><strong>bbox(const point &amp;center, const vector &amp;extents)</strong></td>
<td>Constructor 3</td>
</tr>
<tr>
<td><strong>bbox(const matrix44 &amp;m)</strong></td>
<td>Construct bounding box from matrix44</td>
</tr>
<tr>
<td><strong>point center()</strong> const</td>
<td>Get center of box</td>
</tr>
<tr>
<td><strong>vector extents()</strong> const</td>
<td>Get extents of box</td>
</tr>
<tr>
<td><strong>vector size()</strong> const</td>
<td>Get size of box</td>
</tr>
<tr>
<td><strong>scalar diagonal_size()</strong> const</td>
<td>Get diagonal size of box</td>
</tr>
<tr>
<td><strong>void set(const matrix44 &amp;m)</strong></td>
<td>Set from matrix44</td>
</tr>
<tr>
<td><strong>void set(const point &amp;center, const vector &amp;extents)</strong></td>
<td>Set from center point and extents</td>
</tr>
<tr>
<td><strong>void begin_extend()</strong></td>
<td>Begin extending the box</td>
</tr>
<tr>
<td><strong>void extend(const point &amp;p)</strong></td>
<td>Extend the box</td>
</tr>
<tr>
<td><strong>void extend(const bbox &amp;box)</strong></td>
<td>Extend the box</td>
</tr>
<tr>
<td><strong>void end_extend()</strong></td>
<td>This resets the bounding box size to zero if no extend() method was called after begin_extend()</td>
</tr>
<tr>
<td><strong>void transform(const matrix44 &amp;m)</strong></td>
<td>Transform bounding box</td>
</tr>
<tr>
<td><strong>bool intersects(const bbox &amp;box) const</strong></td>
<td>Check for intersection with axis aligned bounding box</td>
</tr>
<tr>
<td><strong>bool contains(const bbox &amp;box) const</strong></td>
<td>Check if this box completely contains the parameter box</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>bool contains (const point &amp;p) const</code></td>
<td>return true if this box contains the position</td>
</tr>
<tr>
<td>ClipStatus::Type <code>clipstatus (const bbox &amp;other) const</code></td>
<td>check for intersection with other bounding box</td>
</tr>
<tr>
<td>ClipStatus::Type <code>clipstatus (const matrix44 &amp;viewProjection) const</code></td>
<td>check for intersection with projection volume</td>
</tr>
<tr>
<td><code>matrix44 to_matrix44 () const</code></td>
<td>create a matrix which transforms a unit cube to this bounding box</td>
</tr>
<tr>
<td><code>point corner_point (int index) const</code></td>
<td>return one of the 8 corner points</td>
</tr>
<tr>
<td><code>void get_clipplanes (const matrix44 &amp;viewProjection, Util::Array&lt; plane &gt; &amp;outPlanes) const</code></td>
<td>return side planes in clip space</td>
</tr>
</tbody>
</table>
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.

void Math::bbox::end_extend() [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.

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.

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:49 2008
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
Math::float2
Math::float2 Class Reference

#include <float2.h>
Detailed Description

A 2-component float vector class.

(C) 2007 RadonLabs GmbH
## Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float2 ()</td>
<td>default constructor, NOTE: does NOT setup components!</td>
</tr>
<tr>
<td>float2 (scalar x, scalar y)</td>
<td>construct from values</td>
</tr>
<tr>
<td>float2 (const float2 &amp;rhs)</td>
<td>copy constructor</td>
</tr>
<tr>
<td>void operator= (const float2 &amp;rhs)</td>
<td>assignment operator</td>
</tr>
<tr>
<td>float2 operator- () const</td>
<td>flip sign</td>
</tr>
<tr>
<td>void operator+= (const float2 &amp;rhs)</td>
<td>inplace add</td>
</tr>
<tr>
<td>void operator-= (const float2 &amp;rhs)</td>
<td>inplace sub</td>
</tr>
<tr>
<td>void operator *= (scalar s)</td>
<td>inplace scalar multiply</td>
</tr>
<tr>
<td>float2 operator+ (const float2 &amp;rhs) const</td>
<td>add 2 vectors</td>
</tr>
<tr>
<td>float2 operator- (const float2 &amp;rhs) const</td>
<td>subtract 2 vectors</td>
</tr>
<tr>
<td>float2 operator * (scalar s) const</td>
<td>multiply with scalar</td>
</tr>
<tr>
<td>bool operator== (const float2 &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool operator!= (const float2 &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td>void set (scalar x, scalar y)</td>
<td>set content</td>
</tr>
<tr>
<td>scalar &amp; x ()</td>
<td>read/write access to x component</td>
</tr>
<tr>
<td>scalar &amp; y ()</td>
<td>read/write access to y component</td>
</tr>
<tr>
<td>scalar x () const</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>scalar</td>
<td>y</td>
</tr>
<tr>
<td>scalar</td>
<td>length</td>
</tr>
<tr>
<td>scalar</td>
<td>lengthsq</td>
</tr>
<tr>
<td>float2</td>
<td>abs</td>
</tr>
<tr>
<td>bool</td>
<td>any</td>
</tr>
<tr>
<td>bool</td>
<td>all</td>
</tr>
</tbody>
</table>
Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>maximize</code></td>
<td>(const <code>float2</code> &amp;v0, const <code>float2</code> &amp;v1) return vector made up of largest components of 2 vectors</td>
</tr>
<tr>
<td><code>minimize</code></td>
<td>(const <code>float2</code> &amp;v0, const <code>float2</code> &amp;v1) return vector made up of smallest components of 2 vectors</td>
</tr>
<tr>
<td><code>normalize</code></td>
<td>(const <code>float2</code> &amp;v) return normalized version of vector</td>
</tr>
<tr>
<td><code>lt</code></td>
<td>(const <code>float2</code> &amp;v0, const <code>float2</code> &amp;v1) set less-than components to non-zero</td>
</tr>
<tr>
<td><code>le</code></td>
<td>(const <code>float2</code> &amp;v0, const <code>float2</code> &amp;v1) set less-or-equal components to non-zero</td>
</tr>
<tr>
<td><code>gt</code></td>
<td>(const <code>float2</code> &amp;v0, const <code>float2</code> &amp;v1) set greater-than components to non-zero</td>
</tr>
<tr>
<td><code>ge</code></td>
<td>(const <code>float2</code> &amp;v0, const <code>float2</code> &amp;v1) set greater-or-equal components to non-zero</td>
</tr>
</tbody>
</table>
Math::float4
Math::float4 Class Reference

#include <float4.h>

Inheritance diagram for Math::float4:
Detailed Description

The **float4** class implemented on top of the D3DX math functions.

(C) 2007 Radon Labs GmbH

A 4-component float **vector** class. This is the basis class for points and vectors.

(C) 2007 RadonLabs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Math::line
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>void <code>set (const point &amp;startPoint, const point &amp;endPoint)</code></td>
<td>set start and end point</td>
</tr>
<tr>
<td>const point &amp; <code>start () const</code></td>
<td>get start point</td>
</tr>
<tr>
<td>const point &amp; <code>end () const</code></td>
<td>get end point</td>
</tr>
<tr>
<td>const vector &amp; <code>vec () const</code></td>
<td>get vector</td>
</tr>
<tr>
<td>scalar <code>length () const</code></td>
<td>get length</td>
</tr>
<tr>
<td>scalar <code>lengthsq () const</code></td>
<td>get squared length</td>
</tr>
<tr>
<td>scalar <code>distance (const point &amp;p) const</code></td>
<td>minimal distance of point to line</td>
</tr>
<tr>
<td>bool <code>intersect (const line &amp;l, point &amp;pa, point &amp;pb) const</code></td>
<td>intersect with line</td>
</tr>
<tr>
<td>scalar <code>closestpoint (const point &amp;p) const</code></td>
<td>return t of the closest point on the line</td>
</tr>
<tr>
<td>point <code>pointat (scalar t) const</code></td>
<td>return p = b + m*t</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
bool Math::line::intersect(const line & l, 
                           & point & pa,  
                           & point & pb)
```

intersect with **line**

Get line/line intersection. Returns the shortest **line** between two lines.

**Todo:**
: Untested! Replace with simpler code.

```cpp
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*t

```cpp
point Math::line::pointat(scalar t) const [inline]
```

return p = b + m*t

Returns p = b + m * t, given t. Note that the **point** is not on the **line** if 0.0 > t > 1.0
Math::matrix44
Math::matrix44 Class Reference

#include <d3dx9_matrix44.h>
Detailed Description

A `matrix44` class on top of D3DX9 math functions.

(C) 2007 Radon Labs GmbH
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 \textbf{gen} (float x, float y, float z)

*generate noise value*
Math::plane
Math::plane Class Reference

#include <plane.h>
Detailed Description

A plane class on top of D3DX9 math functions.

(C) 2007 Radon Labs GmbH

Nebula's plane class.

(C) 2007 RadonLabs GmbH

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:50 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

Math::point
Math::point Class Reference

#include <point.h>

Inheritance diagram for 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><strong>point ()</strong></td>
<td>Default constructor</td>
</tr>
<tr>
<td><strong>point (scalar x, scalar y, scalar z)</strong></td>
<td>Construct from components</td>
</tr>
<tr>
<td><strong>point (const float4 &amp;rhs)</strong></td>
<td>Construct from float4</td>
</tr>
<tr>
<td><strong>point (const point &amp;rhs)</strong></td>
<td>Copy constructor</td>
</tr>
<tr>
<td><strong>void operator= (const point &amp;rhs)</strong></td>
<td>Assignment operator</td>
</tr>
<tr>
<td><strong>void operator+= (const vector &amp;rhs)</strong></td>
<td>Inplace add vector</td>
</tr>
<tr>
<td><strong>void operator-= (const vector &amp;rhs)</strong></td>
<td>Inplace subtract vector</td>
</tr>
<tr>
<td><strong>point operator+ (const vector &amp;rhs)</strong></td>
<td>Const add point and vector</td>
</tr>
<tr>
<td><strong>point operator- (const vector &amp;rhs)</strong></td>
<td>Const subtract vectors from point</td>
</tr>
<tr>
<td><strong>vector operator- (const point &amp;rhs)</strong></td>
<td>Const subtract point from point into a vector</td>
</tr>
<tr>
<td><strong>bool operator==(const point &amp;rhs)</strong></td>
<td>Const equality operator</td>
</tr>
<tr>
<td><strong>bool operator!=(const point &amp;rhs)</strong></td>
<td>Const inequality operator</td>
</tr>
<tr>
<td><strong>void set (scalar x, scalar y, scalar z)</strong></td>
<td>Const set components</td>
</tr>
</tbody>
</table>
Static Public Member Functions

```c
static point origin ()
return a point at the origin (0, 0, 0)
```

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:50 2008
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><code>polar()</code></td>
<td>the default constructor</td>
</tr>
<tr>
<td><code>polar(scalar t, scalar r)</code></td>
<td>constructor, theta and rho args</td>
</tr>
<tr>
<td><code>polar(const vector &amp;v)</code></td>
<td>constructor, normalized cartesian vector as arg</td>
</tr>
<tr>
<td><code>polar(const polar &amp;src)</code></td>
<td>the copy constructor</td>
</tr>
<tr>
<td><code>void operator=(const polar &amp;rhs)</code></td>
<td>the assignment operator</td>
</tr>
<tr>
<td><code>vector get_cartesian()</code> const</td>
<td>convert to normalized cartesian coords</td>
</tr>
<tr>
<td><code>void set(const polar &amp;p)</code></td>
<td>set to polar object</td>
</tr>
<tr>
<td><code>void set(scalar t, scalar r)</code></td>
<td>set to theta and rho</td>
</tr>
<tr>
<td><code>void set(const vector &amp;v)</code></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 <quaternion.h>
Detailed Description

A **quaternion** class on top of the D3DX9 math functions.

(C) 2007 Radon Labs GmbH

Nebula's **quaternion** class.

(C) 2004 RadonLabs GmbH
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>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rectangle ()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>rectangle (TYPE l, TYPE t, TYPE r, TYPE b)</code></td>
<td>constructor 1</td>
</tr>
<tr>
<td><code>set (TYPE l, TYPE t, TYPE r, TYPE b)</code></td>
<td>set content</td>
</tr>
<tr>
<td><code>inside (TYPE x, TYPE y) const</code></td>
<td>return true if point is inside</td>
</tr>
<tr>
<td><code>width () const</code></td>
<td>return width</td>
</tr>
<tr>
<td><code>height () const</code></td>
<td>return height</td>
</tr>
</tbody>
</table>
Math::sphere
Math::sphere Class Reference

#include <sphere_.h>
Detailed Description

A 3-dimensional sphere.

(C) 2004 RadonLabs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sphere</strong> ()</td>
<td>default constructor</td>
</tr>
<tr>
<td><strong>sphere</strong> (const <strong>point</strong> &amp;p, scalar _r)</td>
<td>pos/radius constructor</td>
</tr>
<tr>
<td><strong>sphere</strong> (scalar _x, scalar _y, scalar _z, scalar _r)</td>
<td>x,y,z,r constructor</td>
</tr>
<tr>
<td><strong>sphere</strong> (const <strong>sphere</strong> &amp;rhs)</td>
<td>copy constructor</td>
</tr>
<tr>
<td>void <strong>set</strong> (const <strong>point</strong> &amp;p, scalar _r)</td>
<td>set position and radius</td>
</tr>
<tr>
<td>void <strong>set</strong> (scalar _x, scalar _y, scalar _z, scalar _r)</td>
<td>set x,y,z, radius</td>
</tr>
<tr>
<td>bool <strong>inside</strong> (const <strong>bbox</strong> &amp;box) const</td>
<td>return true if box is completely inside <strong>sphere</strong></td>
</tr>
<tr>
<td>bool <strong>intersects</strong> (const <strong>sphere</strong> &amp;s) const</td>
<td>check if 2 spheres overlap</td>
</tr>
<tr>
<td>bool <strong>intersects</strong> (const <strong>bbox</strong> &amp;box) const</td>
<td>check if <strong>sphere</strong> intersects box</td>
</tr>
<tr>
<td>bool <strong>intersect_sweep</strong> (const <strong>vector</strong> &amp;va, const <strong>sphere</strong> &amp;sb, const <strong>vector</strong> &amp;vb, scalar &amp;u0, scalar &amp;u1) const</td>
<td>check if 2 moving <strong>sphere</strong> have contact</td>
</tr>
<tr>
<td><strong>rectangle</strong>&lt; scalar &gt; <strong>project_screen_rh</strong> (const <strong>matrix44</strong> &amp;modelView, const <strong>matrix44</strong> &amp;projection, scalar nearZ) const</td>
<td>project <strong>sphere</strong> to screen <strong>rectangle</strong> (right handed coordinate system)</td>
</tr>
<tr>
<td>ClipStatus::Type <strong>clipstatus</strong> (const <strong>bbox</strong> &amp;box) const</td>
<td>get clip status of box against <strong>sphere</strong></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
template <typename T>
bool Math::sphere::inside (const bbox & box) const [inline]
```

return true if box is completely inside `sphere`

Return true if the bounding box is inside the `sphere`.

```cpp
template <typename T>
bool Math::sphere::intersects (const 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

```cpp
template <typename T>
bool Math::sphere::intersect_sweep (const vector & va,
                                     const sphere & sb,
                                     const vector & vb,
                                     scalar & u0,
                                     scalar & u1
                                   ) const
```

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
\uu0\ [out] normalized intro contact
\uu1\ [out] normalized outro contact

\texttt{rectangle< scalar >}
\texttt{Math::sphere::project_screen_rh (\texttt{const matrix44 view},
&
\texttt{const matrix44 projection},
&
\texttt{scalar nearZ })\ const}

Project the \texttt{sphere} (defined in global space) to a screen space \texttt{rectangle}, given the current View and Projection matrices. The method assumes that the \texttt{sphere} is at least partially visible.

\texttt{ClipStatus::Type}
\texttt{Math::sphere::clipstatus (\texttt{const bbox box })\ const [inline]}

get clip status of box against \texttt{sphere}

Get the clip status of a box against this \texttt{sphere}. Inside means: the box is completely inside the \texttt{sphere}.
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

- **transform44 ()**
  
  `transform44()` is a constructor.

- **void setposition (const point &p)**
  
  Set position.

- **const point & getposition () const**
  
  Get position.

- **void setrotate (const quaternion &r)**
  
  Set rotate.

- **const quaternion & getrotate () const**
  
  Get rotate.

- **void setscale (const vector &s)**
  
  Set scale.

- **const vector & getscale () const**
  
  Get scale.

- **void setrotatepivot (const point &p)**
  
  Set optional rotate pivot.

- **const point & getrotatepivot () const**
  
  Get optional rotate pivot.

- **void setscalepivot (const point &p)**
  
  Set optional scale pivot.

- **const point & getscalepivot () const**
  
  Get optional scale pivot.

- **const matrix44 & getmatrix ()**
  
  Get resulting 4x4 matrix.

- **bool isdirty () const**
  
  Return true if the transformation matrix is dirty.
Math::vector
Math::vector Class Reference

#include <vector.h>

Inheritance diagram for Math::vector:
A vector in homogenous space. A vector describes a direction and length in 3d space and always has a w component of 0.0.

(C) 2007 Radon Labs GmbH

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vector()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>vector(scalar x, scalar y, scalar z)</code></td>
<td>construct from components</td>
</tr>
<tr>
<td><code>vector(const float4 &amp;rhs)</code></td>
<td>construct from <code>float4</code></td>
</tr>
<tr>
<td><code>vector(const vector &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>operator=(const vector &amp;rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td><code>operator-()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>operator[](const vector &amp;rhs)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>operator+=()</code></td>
<td>add <code>vector</code> inplace</td>
</tr>
<tr>
<td><code>operator-=</code>(const vector &amp;rhs)</td>
<td>subtract <code>vector</code> inplace</td>
</tr>
<tr>
<td><code>operator*=(scalar s)</code></td>
<td>scale <code>vector</code> inplace</td>
</tr>
<tr>
<td><code>operator+(const vector &amp;rhs)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>operator+(const vector &amp;rhs)</code></td>
<td>add 2 vectors</td>
</tr>
<tr>
<td><code>operator-(const vector &amp;rhs)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>operator-(const vector &amp;rhs)</code></td>
<td>subtract 2 vectors</td>
</tr>
<tr>
<td><code>operator*(scalar s)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>set(scalar x, scalar y, scalar z)</code></td>
<td>scale <code>vector</code></td>
</tr>
</tbody>
</table>

### Equality and Inequality Operators

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>operator==(const vector &amp;rhs)</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>operator!=(const vector &amp;rhs)</code></td>
<td>inequality operator</td>
</tr>
</tbody>
</table>

### Set Components

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>set(scalar x, scalar y, scalar z)</code></td>
<td>set components</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nullvec()</code></td>
<td>return the null vector</td>
</tr>
<tr>
<td><code>upvec()</code></td>
<td>return the standard up vector (0, 1, 0)</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:50 2008
Memory::Heap
Memory::Heap Class Reference

#include <heap.h>
Detailed Description

Implements a private heap.

(C) 2006 Radon Labs GmbH
Memory::Memory
Memory::Memory Class Reference

#include <memory.h>
Detailed Description

Implements a private heap.

(C) 2006 Radon Labs GmbH
Memory::MemoryStatus
Memory::MemoryStatus Struct Reference

#include <win32memory.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::AddAttachment
#include <addattachment.h>
Detailed Description

Attach a graphics entity defined by a resource name to a joint.

(C) 2005 Radon Labs GmbH
Messaging::AddSkin
Messaging::AddSkin Class Reference

#include <addskin.h>
Detailed Description

Makes the given skin visible on a Character3.

(C) 2006 Radon Labs GmbH
Messaging::AnimationHotspotTriggered
Messaging::AnimationHotspotTriggered
Class Reference

#include <animationhotspottriggered.h>
Detailed Description

(C) 2005 Radon Labs GmbH

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:50 2008
Messaging::ApplyImpulseAtPos
Messaging::ApplyImpulseAtPos Class Reference

#include <applyimpulseatpos.h>
Detailed Description

Apply an impulse vector at a position in the global coordinate frame to the physics entity of the game entity.

(C) 2006 Radon Labs GmbH
Messaging::AsyncPort
Messaging::AsyncPort Class Reference

#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. Instead, subclasses of AsyncPort are responsible for creating the handler thread and creating and attaching the handlers to the port.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsyncPort ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~AsyncPort ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>SetName (const Util::String &amp;n)</strong></td>
<td>set the name of the async port (required)</td>
</tr>
<tr>
<td><strong>GetName () const</strong></td>
<td>get the name of the async port</td>
</tr>
<tr>
<td><strong>SetThreadPriority (Threading::Thread::Priority pri)</strong></td>
<td>set optional thread priority</td>
</tr>
<tr>
<td><strong>GetThreadPriority () const</strong></td>
<td>get optional thread priority</td>
</tr>
<tr>
<td>void <strong>SetThreadStackSize</strong> (unsigned int s)</td>
<td>set optional thread stack size</td>
</tr>
<tr>
<td><strong>GetThreadStackSize () const</strong></td>
<td>get optional thread stack size</td>
</tr>
<tr>
<td><strong>SetWaitForMessages</strong> (bool b)</td>
<td>wait-for-messages or run-continuously? (default is wait-for-message)</td>
</tr>
<tr>
<td><strong>GetWaitForMessages () const</strong></td>
<td>get wait-for-message mode</td>
</tr>
<tr>
<td><strong>AttachHandler (const Ptr&lt; Handler &gt; &amp;h)</strong></td>
<td>attach a handler to the port (call before open!)</td>
</tr>
<tr>
<td>virtual <strong>Open ()</strong></td>
<td>open the async port</td>
</tr>
<tr>
<td>virtual <strong>Close ()</strong></td>
<td>close the async port</td>
</tr>
<tr>
<td><strong>IsOpen () const</strong></td>
<td>return true if port is open</td>
</tr>
<tr>
<td><strong>Send</strong> (const Ptr&lt; Message &gt; &amp;msg)</td>
<td>send an asynchronous message to the port</td>
</tr>
<tr>
<td><strong>SendWait</strong> (const Ptr&lt; Message &gt; &amp;msg)</td>
<td>send an asynchronous message to the port</td>
</tr>
</tbody>
</table>
send a message and wait for completion

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void Wait (const Ptr&lt; Message &gt; &amp;msg)</td>
<td>wait for a message to be handled</td>
</tr>
<tr>
<td>bool Peek (const Ptr&lt; Message &gt; &amp;msg)</td>
<td>peek a message whether it has been handled</td>
</tr>
<tr>
<td>void Cancel (const Ptr&lt; Message &gt; &amp;msg)</td>
<td>cancel a pending message</td>
</tr>
<tr>
<td>void Flush ()</td>
<td>wait until all pending messages have been handled</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><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
virtual void onCreateHandlers ()
```

derive in subclass, create and attach handlers from here
Member Function Documentation

```cpp
void Messaging::AsyncPort::AttachHandler(const Ptr<Handler> &h)
attach a handler to the port (call before open!)
Called by OnCreateHandlers() method of subclass to attach a handler to the port.

void Messaging::AsyncPort::Open()
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. This method will first call OnCreateHandlers() and then create the handler thread.
Reimplemented in AsyncHttp::AsyncHttpInterface, and AsyncGraphics::AsyncGraphicsInterface.

void Messaging::AsyncPort::Close()
close the async port
Closes the async port.

void Messaging::AsyncPort::Send(const Ptr<Message> &msg)
send an asynchronous message to the port
Handle an asynchronous message and return immediately. If the caller expects any results from the message he can poll with the AsyncPort::Peak() method, or he may wait for the message to be
handled with the **AsyncPort::Wait()** method.

```cpp
void Messaging::AsyncPort::SendWait ( const Ptr<Message> & msg )
```

send a message and wait for completion

Send an asynchronous message and wait until the message has been handled.

```cpp
void Messaging::AsyncPort::Wait ( const Ptr<Message> & msg )
```

wait for a message to be handled

This method will wait until a message has been handled. If the caller expects any return arguments from the message handling it can use this method to wait for the results.

```cpp
bool Messaging::AsyncPort::Peek ( const Ptr<Message> & msg )
```

peek a message whether it has been handled

This method peeks whether a message has already been handled. If the caller expects any return arguments from the message handling it can use this message to check whether the results are ready using this non-blocking method. The caller can also wait for the results to become ready using the **Wait()** method.

```cpp
void Messaging::AsyncPort::Cancel ( const Ptr<Message> & msg )
```

cancel a pending message
This method will cancel a pending message.

```cpp
void Messaging::AsyncPort::Flush()
```

wait until all pending messages have been handled

This method will wait until ALL pending messages have been handled. Note that this method will be called automatically before the `AsyncPort` shutdown.

```cpp
void Messaging::AsyncPort::OnCreateHandlers() [protected, virtual]
```

derive in subclass, create and attach handlers from here

This message is called by `Open()` when message handlers should be created and attached via `AttachHandler()`. Override this method in a subclass.

Reimplemented in `AsyncHttp::AsyncHttpInterface`, and `AsyncGraphics::AsyncGraphicsInterface`.

```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::Dispatcher
Messaging::Dispatcher Class Reference

#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><code>Dispatcher()</code></td>
<td>Constructor</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 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>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></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</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

```cpp
void Messaging::Dispatcher::HandleMessage (const Ptr<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`.

```cpp
void Messaging::Dispatcher::AttachPort (const Ptr<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 Ptr<Port> &port)
```

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
```

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 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::FadeAnimation
Messaging::FadeAnimation Class Reference

#include <fadeanimation.h>
Detailed Description

Fade an animation on an actor.

(C) 2007 Radon Labs GmbH
Messaging::GetActiveAnimation
Messaging::GetActiveAnimation Class Reference

#include <getactiveanimation.h>
Detailed Description

Get base or overlay animation of an actor.

(C) 2005 Radon Labs GmbH
Messaging::GetAnimationInfo
Messaging::GetAnimationInfo Class Reference

#include <getanimationinfo.h>
Detailed Description

Get animation related information.

(C) 2007 Radon Labs GmbH
Messaging::GetAttachmentEntities
Messaging::GetAttachmentEntities
Class Reference

#include <getattachmententities.h>
Detailed Description

Access a character's attachment (gfx-)entities.

(C) 2005 Radon Labs GmbH
Messaging::GetHotspotTime
#include <gethotspottime.h>
Detailed Description

Return the time of a hotspot

(C) 2007 Radon Labs GmbH
Messaging::GetJointMatrix
Messaging::GetJointMatrix Class Reference

#include <getjointmatrix.h>
Detailed Description

Returns the current transformation of a character joint in world space.

(C) 2006 Radon Labs GmbH
Messaging::GetPhysicsEntity
Messaging::GetPhysicsEntity Class Reference

#include <getphysicsentity.h>
Detailed Description

Returns the Physics entity used by a game entity.

(C) 2006 Radon Labs GmbH
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>Handler ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~Handler ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void Open ()</td>
<td>called once on startup</td>
</tr>
<tr>
<td>virtual void Close ()</td>
<td>called once before shutdown</td>
</tr>
<tr>
<td>bool 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>virtual void DoWork ()</td>
<td>optional &quot;per-frame&quot; DoWork method for continuous handlers</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>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------------------</td>
</tr>
<tr>
<td>bool <code>IsA</code> (const <code>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;</code> <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>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 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 AsyncGraphics::AsyncGraphicsHandler.

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 AsyncGraphics::AsyncGraphicsHandler.

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.

void Messaging::Handler::DoWork ( ) [virtual]

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 `AsyncGraphics::AsyncGraphicsHandler`.

```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

**Messaging::HasAttachment**
Messing::HasAttachment Class Reference

#include <hasattachment.h>
Detailed Description

Checks whether an attachment exists on the entity. The same filtering rules apply as RemAttachment.

(C) 2007 Radon Labs GmbH
Messaging::HideAttachment Class Reference

#include <hideattachment.h>
Detailed Description

Hide one or more attachment graphics entities. This doesn't remove the attachment, just set the graphics entity to hidden! The same filter rules apply as in RemAttachments.

(C) 2007 Radon Labs GmbH
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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Id</strong> ()</td>
<td></td>
</tr>
</tbody>
</table>
*constructor* |
| **bool operator==** (const **Id** &rhs) const |  
*equality operator* |
Messaging::Message
#include <message.h>

Inheritance diagram for Messaging::Message:

- Core::RefCounted
  - Messaging::Message
    - BaseGameFeature::MoveFollow
    - GraphicsFeature::CameraDistance
    - GraphicsFeature::CameraFocus
    - GraphicsFeature::CameraOrbit
    - GraphicsFeature::CameraReset
    - GraphicsFeature::GetGraphicsEntities
    - GraphicsFeature::InputFocus
    - Http::HttpRequest
    - Interface::CopyFile
    - Interface::IOMessage
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

- **Message ()**
  
  *constructor*

- **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*

- **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>const <code>Util::String</code> &amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<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>DumpRefCountingLeaks()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBU_A3_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.
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>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MessageReader</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual void <strong>SetStream</strong></td>
<td><em>Ptr&lt; IO::Stream&gt;</em> &amp;s)</td>
</tr>
<tr>
<td>Message * <strong>ReadMessage</strong></td>
<td>()</td>
</tr>
<tr>
<td>const <strong>GetStream</strong> ()</td>
<td>const <em>Ptr&lt; Stream&gt;</em> &amp;</td>
</tr>
<tr>
<td>bool <strong>HasStream</strong> ()</td>
<td>()</td>
</tr>
<tr>
<td>bool <strong>Eof</strong> ()</td>
<td>()</td>
</tr>
<tr>
<td>virtual bool <strong>Open</strong> ()</td>
<td>()</td>
</tr>
<tr>
<td>virtual void <strong>Close</strong> ()</td>
<td></td>
</tr>
<tr>
<td>bool <strong>IsOpen</strong> ()</td>
<td>()</td>
</tr>
<tr>
<td>int <strong>GetRefCount</strong> ()</td>
<td>()</td>
</tr>
<tr>
<td>void <strong>AddRef</strong> ()</td>
<td>()</td>
</tr>
<tr>
<td>void <strong>Release</strong> ()</td>
<td>()</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong></td>
<td>(const <em>Rtti</em> &amp;rtti) const</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong></td>
<td>(const Util::String &amp;className) const</td>
</tr>
<tr>
<td>bool <strong>IsInstanceOf</strong></td>
<td>(const Util::FourCC &amp;classFourCC) const</td>
</tr>
<tr>
<td>bool <strong>IsA</strong> (const <em>Rtti</em></td>
<td>()</td>
</tr>
<tr>
<td></td>
<td>&amp;rtti) const</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>return true if this object is instance of given class, or a derived class</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool <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>bool <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>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>function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void <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

**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.

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.

bool
IO::StreamReader::HasStream( ) const [inherited]

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.
Messaging::MessageWriter
#include <messagewriter.h>

Inheritance diagram for Messaging::MessageWriter:

```
Core::RefCounted
   |
   V
IO::StreamWriter
   |
   V
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>virtual void <strong>SetStream</strong> (const Ptr&lt; IO::Stream &gt; &amp;s)</td>
<td>set stream to write to</td>
</tr>
<tr>
<td>void <strong>WriteMessage</strong> (const Ptr&lt; Message &gt; &amp;msg)</td>
<td>write a complete message to the stream</td>
</tr>
<tr>
<td>const Ptr&lt; Stream &gt; &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>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>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>
</tbody>
</table>

IO::Stream
Message
Util::String
Util::FourCC
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
**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::MoveDirection
Messaging::MoveDirection Class Reference

#include <movedirection.h>
Detailed Description

A MoveDirection message. Expected behaviour is that the entity starts to move into the specified direction. The direction vector can be defined as camera relative or absolute. The velocity settings should be interpreted as a factor.

(C) 2007 Radon Labs GmbH
Messaging::MoveGoto
Messaging::MoveGoto Class Reference

#include <movegoto.h>
**Detailed Description**

A highlevel **MoveGoto Messaging**. The expected behaviour is that an entity should move to the target point defined by the move goto **Messaging** and stop there. If possible, the entity should steer around obstacles and use pathfinding to reach the target point.

(C) 2005 Radon Labs GmbH
Messaging::MoveRotate
#include <moverotate.h>
Detailed Description

**Commands** an entity rotate around the y-axis for a new heading.

(C) 2006 Radon Labs GmbH
Messaging::MoveSetVelocity
Messaging::MoveSetVelocity Class Reference

#include <movesetvelocity.h>
Detailed Description

Set the (relative) linear velocity of an entity between 0.0 and 1.0. The actual resulting velocity also depends on the MaxVelocity attribute attached to the entity.

(C) 2005 Radon Labs GmbH
Messaging::MoveStop
Messaging::MoveStop Class Reference

#include <movestop.h>
Detailed Description

A \textbf{MoveStop} message. The expected behaviour is that an entity which receives this message stops immediately.

(C) 2007 Radon Labs GmbH
Messaging::MoveTurn
Messaging::MoveTurn Class Reference

#include <moveturn.h>
Detailed Description

**Commands** an entity to turn into the specified direction defined by a 3d vector. The direction vector can be absolute or camera relative.
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>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void SetupAcceptedMessages ()</td>
<td>override to register accepted messages</td>
<td></td>
</tr>
<tr>
<td>void AttachHandler (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>attach a message handler to the port</td>
<td></td>
</tr>
<tr>
<td>void RemoveHandler (const Ptr&lt; Handler &gt; &amp;h)</td>
<td>remove a message handler from the port</td>
<td></td>
</tr>
<tr>
<td>SizeT GetNumHandlers () const</td>
<td>return number of handlers attached to the port</td>
<td></td>
</tr>
<tr>
<td>const Ptr&lt; Handler &gt; &amp; GetHandlerAtIndex (IndexT i) const</td>
<td>get a message handler by index</td>
<td></td>
</tr>
<tr>
<td>virtual void Send (const Ptr&lt; Message &gt; &amp;msg)</td>
<td>send a message to the port</td>
<td></td>
</tr>
<tr>
<td>const Util::Array &lt; const Id * &gt; &amp; GetAcceptedMessages () const</td>
<td>get the array of accepted messages (sorted)</td>
<td></td>
</tr>
<tr>
<td>bool AcceptsMessage (const Id &amp;msgId) const</td>
<td>return true if port accepts this msg</td>
<td></td>
</tr>
<tr>
<td>virtual void HandleMessage (const Ptr&lt; Messaging::Message &gt; &amp;msg)</td>
<td>handle a single accepted message</td>
<td></td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
<td></td>
</tr>
<tr>
<td>void AddRef ()</td>
<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>
<td></td>
</tr>
<tr>
<td>bool IsInstanceOf (const Rtti &amp;rtti) const</td>
<td>return true if this object is instance of given class</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>
<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>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><code>IsA</code> (const <code>Rtti</code> &amp;rtti) const</td>
<td></td>
</tr>
<tr>
<td>------</td>
<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>
<td></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>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
void RegisterMessage (const Id &msgId)

register a single accepted message
```
### Member Function Documentation

**void Messaging::Port::AttachHandler**

```cpp
class Messaging::Port
{
public:
    void AttachHandler(const Ptr<Handler> &h);
};
```

attach a message handler to the port

Attach a message handler to the port.

**void Messaging::Port::RemoveHandler**

```cpp
class Messaging::Port
{
public:
    void RemoveHandler(const Ptr<Handler> &h);
};
```

remove a message handler from the port

Remove a message handler from the port.

**void Messaging::Port::Send**

```cpp
class Messaging::Port
{
public:
    void 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.

**void Messaging::Port::HandleMessage**

```cpp
class Messaging::Port
{
public:
    void HandleMessage(const Ptr<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()`.
Reimplemented in **Messaging::Dispatcher**.

```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::RemAttachment
Messaging::RemAttachment Class Reference

#include <remattachment.h>

====================================================================
Detailed Description

Remove an attachment from a joint.

(C) 2005 Radon Labs GmbH
Messaging::RemSkin
#include <remskin.h>
Detailed Description

Makes the given skin invisible on a Character3.

(C) 2006 Radon Labs GmbH
Messaging::SetAnimation
Messaging::SetAnimation Class Reference

#include <setanimation.h>
Detailed Description

Set base or overlay animation on an actor.

(C) 2005 Radon Labs GmbH
Messaging::SetFadeAnimationMix
Messaging::SetFadeAnimationMix
Class Reference

#include <setfadeanimationmix.h>
Detailed Description

Set fade animation on an actor.

(C) 2007 Radon Labs GmbH
Messaging::SetTransform
Messaging::SetTransform Class Reference

#include <settransform.h>
Detailed Description

Set the complete transform of a entity, including the physics transform. Use not per Frame, the physics won't like it!

(C) 2007 Radon Labs GmbH
Messaging::ShowAttachment
Messaging::ShowAttachment Class Reference

#include <showattachment.h>
Detailed Description

Hide one or more attachment graphics entities. This doesn't remove the attachment, just set the graphics entity to hidden! The same filter rules apply as in RemAttachments.

(C) 2007 Radon Labs GmbH
Messaging::UpdateAttachments
Messaging::UpdateAttachments Class Reference

#include <updateattachments.h>
Detailed Description

Update a character's attachments.

(C) 2006 Radon Labs GmbH
Messaging::UpdateTransform
Messaging::UpdateTransform Class Reference

#include <updatetransform.h>
Detailed Description

Updates the transform of a entity, does not set the physics transform. All property that need to update when entity transformation changes need listen to this message.

To set the transformation of a entity (including the physics) use the **SetTransform** message.

(C) 2007 Radon Labs GmbH
Models::BinaryModelReader
Models::BinaryModelReader Class Reference

#include <binarymodelreader.h>

Inheritance diagram for Models::BinaryModelReader:
Detailed Description

Implements the ModelReader for the binary file format.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BinaryModelReader()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~BinaryModelReader()</code></td>
<td>Destructor</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>virtual bool FillModel()</code></td>
<td>Parse resource and build model hierarchy</td>
</tr>
<tr>
<td><code>void SetModelResId(const Resources::ResourceId &amp;resId)</code></td>
<td>Set the read cursor to the first Model in the stream</td>
</tr>
<tr>
<td><code>const Resources::ResourceId &amp; GetModelResId()</code></td>
<td>Get the model resource id</td>
</tr>
<tr>
<td><code>void SetModel(const Ptr&lt;Model&gt; &amp;model)</code></td>
<td>Get attributes of current model</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()</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>bool Eof()</code></td>
<td>Return true if the stream has reached EOF</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>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>
Member Function Documentation

void Models::ModelReader::SetModelResId (const Resources::ResourceId & resId) [inherited]

set the read cursor to the first Model in the stream

set the read cursor to the next Model in the stream set model resource id (some file format's don't provide their own)

void Models::ModelReader::SetModel (const Ptr<Model> & model) [inline, inherited]

get attributes of current model

set the read cursor to the first ModelNode in the current Model set the read cursor to the next ModelNode in the current Model get the model node name get model node class get the name of the parent node get attributes of current model node set model for filling up with modelnodes and attributes

void IO::StreamReader::SetStream (const Ptr<Stream> & s) [inherited]

set stream to read from

Attaches the reader to a stream. This will imcrement the refcount of the stream.

Reimplemented in Messaging::MessageReader.

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.

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

dump 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::BinaryModelWriter
Models::BinaryModelWriter Class Reference

#include <binarymodelwriter.h>

Inheritance diagram for Models::BinaryModelWriter:
Detailed Description

Implements the ModelWriter for the binary file format.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BinaryModelWriter ()</strong></td>
<td>virtual constructor</td>
</tr>
<tr>
<td><strong>~BinaryModelWriter ()</strong></td>
<td>virtual destructor</td>
</tr>
<tr>
<td><strong>Util::String GetFileExtension () const</strong></td>
<td>virtual get the file extension used by the writer</td>
</tr>
<tr>
<td><strong>Open ()</strong></td>
<td>virtual begin reading from the stream</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>virtual end reading from the stream</td>
</tr>
<tr>
<td><strong>BeginModel (const Ptr&lt; Model &gt; &amp;model)</strong></td>
<td>virtual begin writing a new Model</td>
</tr>
<tr>
<td><strong>WriteModelAttrs (const Attr::AttributeContainer &amp;attrs)</strong></td>
<td>virtual write model attributes</td>
</tr>
<tr>
<td><strong>EndModel ()</strong></td>
<td>virtual end writing current Model</td>
</tr>
<tr>
<td><strong>BeginModelNode (const Ptr&lt; ModelNode &gt; &amp;modelNode)</strong></td>
<td>virtual begin writing a new ModelNode</td>
</tr>
<tr>
<td><strong>WriteModelNodeAttrs (const Attr::AttributeContainer &amp;attrs)</strong></td>
<td>virtual write mode node attributes</td>
</tr>
<tr>
<td><strong>EndModelNode ()</strong></td>
<td>virtual end writing current ModelNode</td>
</tr>
<tr>
<td><strong>SetStream (const Ptr&lt; Stream &gt; &amp;s)</strong></td>
<td>void set stream to write to</td>
</tr>
<tr>
<td><strong>GetStream () const</strong></td>
<td>const get currently set stream</td>
</tr>
<tr>
<td><strong>HasStream () const</strong></td>
<td>bool return true if a stream is set</td>
</tr>
<tr>
<td><strong>IsOpen () const</strong></td>
<td>bool return true if currently open</td>
</tr>
<tr>
<td>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>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>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 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.
Models::CharacterNode
Models::CharacterNode Class Reference

#include <characternode.h>

Inheritance diagram for Models::CharacterNode:
Detailed Description

A **TransformNode** which describes a skinned shape. Since the **CharacterNode** is derived from the **TransformNode** (which in turn is derived from the **TransformNode**), a **CharacterNode** contains all the necessary information for rendering a skinned mesh.

It supports the use of variations and skinlists for multi exchangeable animated skins and textures (all the nebula2 character3 functionalities).

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CharacterNode ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~CharacterNode ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>GetResourceState ()</strong> const</td>
<td>Get overall state of contained resources (Initial, Loaded, Pending, Failed, Cancelled)</td>
</tr>
<tr>
<td><strong>GetNumSkins ()</strong> const</td>
<td>Get number of skin infos</td>
</tr>
<tr>
<td><strong>GetSkinIndexByName (const Util::String &amp;n) const</strong></td>
<td>Get skin index by name</td>
</tr>
<tr>
<td><strong>GetSkinInfoAt (IndexT i) const</strong></td>
<td>Get skin info by index</td>
</tr>
<tr>
<td><strong>GetSkinInfoArray () const</strong></td>
<td>Get whole skin info array</td>
</tr>
<tr>
<td><strong>CreateNodeInstance () const</strong></td>
<td>Create a model node instance</td>
</tr>
<tr>
<td><strong>LoadResources ()</strong></td>
<td>Called when resources should be loaded</td>
</tr>
<tr>
<td><strong>UnloadResources ()</strong></td>
<td>Called when resources should be unloaded</td>
</tr>
<tr>
<td><strong>SetAnim (const Util::String &amp;filename)</strong></td>
<td>Set name of an anim resource</td>
</tr>
<tr>
<td><strong>GetAnim () const</strong></td>
<td>Get name of an anim resource</td>
</tr>
<tr>
<td><strong>SetVariationsUri (const Util::String &amp;uri)</strong></td>
<td>Set variations animation resource</td>
</tr>
<tr>
<td><strong>GetVariationsUri () const</strong></td>
<td>Get variations animation resource</td>
</tr>
<tr>
<td><strong>BeginVariations (SizeT num)</strong></td>
<td>Begin defining variations</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>void SetVariation (IndexT varIndex, IndexT animGroupIndex, const Util::String &amp;varName)</td>
<td>set a variation</td>
</tr>
<tr>
<td>void EndVariations ()</td>
<td>finish defining variations</td>
</tr>
<tr>
<td>SizeT GetNumVariations () const</td>
<td>get number of variations</td>
</tr>
<tr>
<td>const Util::String &amp; GetVariationNameAt (IndexT i) const</td>
<td>get variation name at index</td>
</tr>
<tr>
<td>const Util::Array &lt; Char::CharJoint &gt; &amp; GetVariationJointsAt (IndexT i) const</td>
<td>get variation joints at index</td>
</tr>
<tr>
<td>IndexT GetVariationIndexByName (const Util::String &amp;n) const</td>
<td>get variation index by name</td>
</tr>
<tr>
<td>void BeginClips (int numClips)</td>
<td>begin adding clips</td>
</tr>
<tr>
<td>void SetClip (int clipIndex, int animGroupIndex, const Util::String &amp;clipName)</td>
<td>add an animation clip</td>
</tr>
<tr>
<td>void EndClips ()</td>
<td>finish adding clips</td>
</tr>
<tr>
<td>int GetNumClips () const</td>
<td>get number of clips in the animation</td>
</tr>
<tr>
<td>const Anim::AnimClip &amp; GetClipAt (int clipIndex) const</td>
<td>get clip at index</td>
</tr>
<tr>
<td>IndexT GetClipIndexByName (const Util::String &amp;name) const</td>
<td>get clip by name</td>
</tr>
<tr>
<td>Timing::Time GetClipDuration (int index) const</td>
<td>get clip duration</td>
</tr>
<tr>
<td>const Ptr &lt; Char::CharacterSet &gt; &amp; GetCharacterSet () const</td>
<td>get character set</td>
</tr>
<tr>
<td>void BeginJoints (int numJoints)</td>
<td>begin adding joints</td>
</tr>
<tr>
<td>SetJoint (int index, int parentJointIndex,</td>
<td>set a variation</td>
</tr>
</tbody>
</table>
void const Math::vector &poseTranslate, const Math::quaternion &poseRotate, const Math::vector &poseScale, const Util::String &name)

add a joint to the skeleton

void EndJoints ()
finish adding joints

int GetNumJoints ()
get number of joints in skeleton

void GetJoint (int index, int &parentJointIndex, Math::vector &poseTranslate, Math::quaternion &poseRotate, Math::vector &poseScale, Util::String &name)
get joint attributes

const Ptr<Anim::ManagedAnimation> & GetManagedAnimation () const
get managed animation

const Ptr<Char::Character> & GetCharacter () const
get character

void SetPosition (const Math::point &p)
set position

const Math::point & GetPosition () const
get position

void SetRotate (const Math::quaternion &r)
set rotate quaternion

const Math::quaternion & GetRotate () 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

SetScalePivot (const Math::point &p)
<table>
<thead>
<tr>
<th>void</th>
<th>set scale pivot</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Math::point &amp;</td>
<td>GetScalePivot () const</td>
</tr>
<tr>
<td>virtual void</td>
<td>ApplySharedState ()</td>
</tr>
<tr>
<td>apply state shared by all my ModelNodeInstances</td>
<td></td>
</tr>
<tr>
<td>const Util::Atom<a href="">Util::String</a> &amp;</td>
<td>GetName () const</td>
</tr>
<tr>
<td>get model node name</td>
<td></td>
</tr>
<tr>
<td>ModelNodeType::Code</td>
<td>GetType () const</td>
</tr>
<tr>
<td>get the ModelNodeType</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>HasParent () const</td>
</tr>
<tr>
<td>return true if node has a parent</td>
<td></td>
</tr>
<tr>
<td>const Ptr&lt;ModelNode&gt; &amp;</td>
<td>GetParent () const</td>
</tr>
<tr>
<td>get parent node</td>
<td></td>
</tr>
<tr>
<td>const Util::Array&lt;Ptr&lt;ModelNode&gt;&gt; &amp;</td>
<td>GetChildren () const</td>
</tr>
<tr>
<td>get child nodes</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>IsAttachedToModel () const</td>
</tr>
<tr>
<td>return true if currently attached to a Model</td>
<td></td>
</tr>
<tr>
<td>const Ptr&lt;Model&gt; &amp;</td>
<td>GetModel () const</td>
</tr>
<tr>
<td>get model this node is attached to</td>
<td></td>
</tr>
<tr>
<td>const Math::bbox &amp;</td>
<td>GetBoundingBox () const</td>
</tr>
<tr>
<td>get bounding box of model node</td>
<td></td>
</tr>
<tr>
<td>const Attr::AttributeContainer &amp;</td>
<td>GetAttrs () const</td>
</tr>
<tr>
<td>read access to model node attributes</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>HasAttr (const Attr::AttrId &amp;attrId) const</td>
</tr>
<tr>
<td>check if model node attribute exists</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>SetAttr (const Attr::Attribute &amp;attr)</td>
</tr>
<tr>
<td>set generic model node attribute</td>
<td></td>
</tr>
<tr>
<td>const Attr::Attribute &amp;</td>
<td>GetAttr (const Attr::AttrId &amp;attrId) const</td>
</tr>
<tr>
<td>get generic model node attribute</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>SetBool (const Attr::BoolAttrId &amp;attrId, bool val)</td>
</tr>
<tr>
<td>set bool value</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>GetBool (const Attr::BoolAttrId &amp;attrId) const</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>void</td>
<td><code>SetFloat</code> (const Attr::FloatAttrId &amp;attrId, float val)</td>
</tr>
<tr>
<td>float</td>
<td><code>GetFloat</code> (const Attr::FloatAttrId &amp;attrId) const</td>
</tr>
<tr>
<td>void</td>
<td><code>SetInt</code> (const Attr::IntAttrId &amp;attrId, int val)</td>
</tr>
<tr>
<td>int</td>
<td><code>GetInt</code> (const Attr::IntAttrId &amp;attrId) const</td>
</tr>
<tr>
<td>void</td>
<td><code>SetString</code> (const Attr::StringAttrId &amp;attrId, const Util::String &amp;val)</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><code>GetString</code> (const Attr::StringAttrId &amp;attrId) const</td>
</tr>
<tr>
<td>void</td>
<td><code>SetFloat4</code> (const Attr::Float4AttrId &amp;attrId, const Math::float4 &amp;val)</td>
</tr>
<tr>
<td>Math::float4</td>
<td><code>GetFloat4</code> (const Attr::Float4AttrId &amp;attrId) const</td>
</tr>
<tr>
<td>void</td>
<td><code>SetMatrix44</code> (const Attr::Matrix44AttrId &amp;attrId, const Math::matrix44 &amp;val)</td>
</tr>
<tr>
<td>const Math::matrix44 &amp;</td>
<td><code>GetMatrix44</code> (const Attr::Matrix44AttrId &amp;attrId) const</td>
</tr>
<tr>
<td>void</td>
<td><code>SetGuid</code> (const Attr::GuidAttrId &amp;attrId, const Util::Guid &amp;guid)</td>
</tr>
<tr>
<td>const Util::Guid &amp;</td>
<td><code>GetGuid</code> (const Attr::GuidAttrId &amp;attrId) const</td>
</tr>
<tr>
<td>void</td>
<td><code>SetBlob</code> (const Attr::BlobAttrId &amp;attrId, const Util::Blob &amp;blob)</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const Util::Blob &amp; GetBlob (const Attr::BlobAttrId &amp;attrId)</code></td>
<td>set blob value</td>
</tr>
<tr>
<td><code>void SetName (const Util::Atom&lt; Util::String &gt; &amp;n)</code></td>
<td>set model node name</td>
</tr>
<tr>
<td><code>void SetType (ModelNodeType::Code t)</code></td>
<td>set <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>void AddChild (const Ptr&lt;ModelNode&gt; &amp;c)</code></td>
<td>add a child node</td>
</tr>
<tr>
<td><code>void 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>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><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></td>
</tr>
</tbody>
</table>
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><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>bool</td>
<td><code>LoadVariation()</code></td>
<td>load variation</td>
</tr>
<tr>
<td><code>Util::Array</code>&lt;<code>Char::CharJoint&gt;</code></td>
<td><code>EvaluateVariation</code> (const <code>Ptr&lt;Anim::Animation&gt;</code> &amp;varAnim, <code>IndexT</code> groupIndex)</td>
<td>extract skeleton variation from a variation clip</td>
</tr>
<tr>
<td>void</td>
<td><code>SetupSkinInfos()</code></td>
<td>setup skin infos from child objects</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>LoadFromAttrs</code> (const <code>Attr::AttributeContainer</code> &amp;attrs)</td>
<td>called to initialize from attributes</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>SaveToAttrs</code> (<code>Attr::AttributeContainer</code> &amp;attrs)</td>
<td>called to save state back to attributes</td>
</tr>
<tr>
<td>void</td>
<td><code>SetBoundingBox</code> (const <code>Math::bbox</code> &amp;b)</td>
<td>set bounding box</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnAttachToModel</code> (const <code>Ptr&lt;Model&gt;</code> &amp;model)</td>
<td>called when attached to model node</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnRemoveFromModel</code> ()</td>
<td>called when removed from model node</td>
</tr>
<tr>
<td>const <code>Util::Array</code>&lt;<code>Ptr&lt;ModelNodeInstance&gt;</code>&gt; &amp;</td>
<td><code>GetVisibleModelNodeInstances</code> ([<code>ModelNodeType::Code</code> t]) const</td>
<td>get visible model node instances</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::CharacterNode::BeginVariations(SizeT num)
```

begin defining variations

Begin defining variations.

```cpp
void Models::CharacterNode::SetVariation(IndexT varIndex,
                                          IndexT animGroupIndex,
                                          const Util::String& varName)
```

set a variation

Define a variation.

```cpp
void Models::CharacterNode::EndVariations()
```

finish defining variations

End defining variations.

```cpp
void Models::CharacterNode::BeginJoints(int numJoints)[inline]
```

begin adding joints

Begin configuring the joint skeleton.

```cpp
void Models::CharacterNode::SetJoint(int jointIndex,
                                      int parentJointIndex,
                                      const Math::vector& poseTranslate,
                                      const Math::quaternion poseRotate,
                                      ...)
```

&
const
Math::vector & poseScale,
const Util::String & name
)
)[inline]

add a joint to the skeleton

Add a joint to the joint skeleton.

void
Models::CharacterNode::EndJoints( ) [inline]

finish adding joints

Finish adding joints to the joint skeleton.

int
Models::CharacterNode::GetNumJoints( ) [inline]

get number of joints in skeleton

Get number of joints in joint skeleton.

void
Models::CharacterNode::GetJoint( int & index,
int & parentJointIndex,
Math::vector & poseTranslate,
Math::quaternion & poseRotate,
Math::vector & poseScale,
Util::String & name
)[inline]

get joint attributes

Get joint attributes.

void
Models::CharacterNode::SetupSkinInfos( ) [protected]

setup skin infos from child objects
Setup the skin info table from my child skin objects.

```cpp
void Models::ModelNode::ApplySharedState() [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 `Models::ShapeNode`, `Models::SkinShapeNode`, and `Models::StateNode`.

```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

Increment 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.
Models::CharacterNodeInstance
Models::CharacterNodeInstance Class Reference

#include <characternodeinstance.h>

Inheritance diagram for Models::CharacterNodeInstance:
Detailed Description

The **CharacterNodeInstance** actually renders a skinned shape, and holds all the necessary per-instance state to do so.

It knows the selected visible skins an active textures for actual rendering.

(C) 2007 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>~CharacterNodeInstance()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>Update()</code></td>
<td>Perform per-frame updates</td>
</tr>
<tr>
<td><code>GetCharacter() const</code></td>
<td>Get character</td>
</tr>
<tr>
<td><code>GetCharacterSet() const</code></td>
<td>Get character set</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() const</code></td>
<td>Get position</td>
</tr>
<tr>
<td><code>SetRotate(const Math::quaternion &amp;r)</code></td>
<td>Set rotate quaternion</td>
</tr>
<tr>
<td><code>GetRotate() const</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() const</code></td>
<td>Get rotate pivot</td>
</tr>
<tr>
<td><code>setScalePivot(const Math::point &amp;p)</code></td>
<td>Set scale pivot</td>
</tr>
<tr>
<td><code>getScalePivot()</code></td>
<td>Get scale pivot</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetLocalTransform()</code></td>
<td>get resulting local transform matrix in local parent space</td>
</tr>
</tbody>
</table>
| `GetModelTransform()` | get model space transform (valid after `Update()`)
<p>| <code>Render()</code> | perform rendering |
| <code>HasParent()</code> | return true if node has a parent |
| <code>GetParent()</code> | get parent node |
| <code>GetChildren()</code> | get child nodes |
| <code>IsAttachedToModelInstance()</code> | return true if attached to <code>ModelInstance</code> |
| <code>GetModelInstance()</code> | get the <code>ModelInstance</code> we are attached to |
| <code>GetModelNode()</code> | get the <code>ModelNode</code> we’re associated with |
| <code>SetVisible(bool b)</code> | set visible, used by charactersystem |
| <code>IsVisible()</code> | is visible |
| <code>GetRefCount()</code> | get the current refcount |
| <code>AddRef()</code> | increment refcount by one |
| <code>Release()</code> | decrement refcount and destroy object if refcount is zero |
| <code>IsInstanceOf(Rtti &amp;rtti)</code> | return true if this object is instance of given class |
| <code>IsInstanceOf(Util::String &amp;className)</code> | return true if this object is instance of given class by string |
| <code>IsInstanceOf(Util::FourCC)</code> | return true if this object is instance of given class by string |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsA(const Rtti &amp;rtti) const</code></td>
<td>return true if this object is instance of given class by Rtti.</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,</td>
</tr>
<tr>
<td></td>
<td>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,</td>
</tr>
<tr>
<td></td>
<td>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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OnAttachToModelInstance</code> (const <code>Ptr&lt; ModelInstance &gt;</code> &amp;inst, const <code>Ptr&lt; ModelNode &gt;</code> &amp;node, const <code>Ptr&lt; ModelNodeInstance &gt;</code> &amp;parentNodeInst)</td>
<td>called when attached to <code>ModelInstance</code></td>
</tr>
<tr>
<td><code>OnRemoveFromModelInstance</code> ()</td>
<td>called when removed from <code>ModelInstance</code></td>
</tr>
<tr>
<td><code>RenderDebug ()</code></td>
<td>render node specific debug shape</td>
</tr>
<tr>
<td><code>ValidateCharacter ()</code></td>
<td>validate character</td>
</tr>
<tr>
<td><code>SetParent (const </code>Ptr&lt; ModelNodeInstance &gt;<code> &amp;p)</code></td>
<td>set parent node</td>
</tr>
<tr>
<td><code>AddChild (const </code>Ptr&lt; ModelNodeInstance &gt;<code> &amp;c)</code></td>
<td>add a child node</td>
</tr>
<tr>
<td><code>OnNotifyVisible (IndexT frameIndex)</code></td>
<td>notify that we are visible</td>
</tr>
<tr>
<td><code>SetChildrenVisibility (ModelNodeInstance *parent, bool b)</code></td>
<td>set visible flag of children</td>
</tr>
</tbody>
</table>
Member Function Documentation

```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`.

```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 `Models::ParticleSystemNodeInstance`, `Models::ShapeNodeInstance`, and `Models::SkinShapeNodeInstance`.

```cpp
void Models::ModelNodeInstance::SetVisible(bool b) [inline, virtual, inherited]
```

set visible, used by charactersystem

Set visibility of node and its children, should not be call per frame!

Reimplemented in `Models::SkinShapeNodeInstance`.

```cpp
void Models::ModelNodeInstance::OnNotifyVisible(IndexT frameIndex) [protected, virtual, inherited]
```
notify that we are visible

This method is called from the NotifyVisible() method of our ModelInstance object. If the ModelNodeInstance provides something renderable it should respond by adding itself as visible node instance to its model node.

Reimplemented in Models::ParticleSystemNodeInstance, Models::ShapeNodeInstance, and Models::SkinShapeNodeInstance.

```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::ManagedModel
Models::ManagedModel Class Reference

#include <managedmodel.h>

Inheritance diagram for Models::ManagedModel:
Detailed Description

A specialized managed resource for **Models**.

(C) 2007 Radon Labs GmbH
**Public Types**

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>Priority</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>priority levels</em></td>
</tr>
</tbody>
</table>
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const <code>Ptr&lt; Model &gt;</code> &amp; <strong>GetModel()</strong> const</td>
<td>get contained model resource</td>
</tr>
<tr>
<td>void <strong>ClearRenderStats()</strong></td>
<td>clear render statistics</td>
</tr>
<tr>
<td>void <strong>UpdateRenderStats(const Math::float2 &amp;screenSpaceSize)</strong></td>
<td>update render statistics (called by client)</td>
</tr>
<tr>
<td>void <strong>SetResourceld(const Resourceld &amp;id)</strong></td>
<td>set resource id</td>
</tr>
<tr>
<td>const <code>Resourceld</code> &amp; <strong>GetResourceld()</strong> const</td>
<td>get resource id</td>
</tr>
<tr>
<td>void *<em>SetResourceType(const Core::Rtti <em>rtti)</em></em></td>
<td>set contained resource type</td>
</tr>
<tr>
<td>const <code>Core::Rtti</code> * <strong>GetResourceType()</strong> const</td>
<td>get contained resource type</td>
</tr>
<tr>
<td>void <strong>IncrClientCount()</strong></td>
<td>increment client count</td>
</tr>
<tr>
<td>void <strong>DecrClientCount()</strong></td>
<td>decrement client count</td>
</tr>
<tr>
<td><code>SizeT</code> <strong>GetClientCount()</strong> const</td>
<td>get current client count</td>
</tr>
<tr>
<td><code>SizeT</code> <strong>GetRenderCount()</strong> const</td>
<td>get render count for this frame (number of calls to <strong>UpdateRenderStats()</strong></td>
</tr>
<tr>
<td>const <code>Math::float2</code> &amp; <strong>GetMaxScreenSpaceSize()</strong> const</td>
<td>get maximum screen space size this frame</td>
</tr>
<tr>
<td>void <strong>SetPriority(Priority p)</strong></td>
<td>set current priority</td>
</tr>
<tr>
<td><code>Priority</code> <strong>GetPriority()</strong> const</td>
<td>get the current priority</td>
</tr>
<tr>
<td><code>Resource::State</code> <strong>getState()</strong> const</td>
<td>get current resource loading state (Initial, Pending, Loaded, Failed, Cancelled)</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Resource &gt;</code> &amp; <strong>GetResource()</strong> const</td>
<td>get resource</td>
</tr>
</tbody>
</table>
void Clear ()

clear the contained resource

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

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 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).
```

```cpp
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.
```

```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::Model
Models::Model Class Reference

#include <model.h>

Inheritance diagram for Models::Model:

```
Core::RefCounted

Resources::Resource

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>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~Model ()</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>Unload ()</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td>void <strong>LoadResources ()</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Load node resources (meshes, textures, shaders, ...)</td>
</tr>
<tr>
<td>void <strong>UnloadResources ()</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Unload node resources</td>
</tr>
<tr>
<td><strong>State</strong> <strong>GetResourceState () const</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Get the overall resource loading state (Initial -&gt; Pending -&gt; Loaded/Failed/Cancelled)</td>
</tr>
<tr>
<td>void <strong>SetBoundingBox (const Math::bbox &amp;b)</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Set the model's local bounding box</td>
</tr>
<tr>
<td>const Math::bbox &amp; <strong>GetBoundingBox () const</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Get the model's local bounding box</td>
</tr>
<tr>
<td>bool <strong>HasNode (const Util::Atom&lt; Util::String &gt; &amp;name) const</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Check if a ModelNode exists</td>
</tr>
<tr>
<td>const Ptr&lt; ModelNode &gt; &amp; <strong>LookupNode (const Util::Atom&lt; Util::String &gt; &amp;name) const</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Lookup a ModelNode in the Model</td>
</tr>
<tr>
<td>void <strong>AttachNode (const Ptr&lt; ModelNode &gt; &amp;node)</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Attach a model node to the Model</td>
</tr>
<tr>
<td>void <strong>RemoveNode (const Ptr&lt; ModelNode &gt; &amp;node)</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Remove a model node from the Model</td>
</tr>
<tr>
<td>const Util::Array&lt; Ptr&lt; ModelNode &gt; &gt; &amp; <strong>GetNodes () const</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Access to model nodes</td>
</tr>
<tr>
<td><strong>Ptr&lt; ModellInstance &gt;</strong> <strong>CreateInstance ()</strong>&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Create a ModellInstance of the Model</td>
</tr>
</tbody>
</table>

---

<sup>1</sup> Indicates that the function is part of the Model class.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Util::Array &amp; GetInstances () const</td>
<td>get all attached model instances</td>
</tr>
<tr>
<td>bool LoadFromAttrs (const Attr::AttributeContainer &amp;attrs)</td>
<td>load model attributes from attribute set</td>
</tr>
<tr>
<td>bool SaveToAttrs (Attr::AttributeContainer &amp;attrs)</td>
<td>save model attributes to attribute set</td>
</tr>
<tr>
<td>const Attr::AttributeContainer &amp; GetAttrs () const</td>
<td>read access to model attributes</td>
</tr>
<tr>
<td>bool HasAttr (const Attr::AttrId &amp;attrId) const</td>
<td>check if model attribute exists</td>
</tr>
<tr>
<td>void SetAttr (const Attr::Attribute &amp;attr)</td>
<td>set generic model attribute</td>
</tr>
<tr>
<td>const Attr::Attribute &amp; GetAttr (const Attr::AttrId &amp;attrId) const</td>
<td>get generic model attribute</td>
</tr>
<tr>
<td>void SetBool (const Attr::BoolAttrId &amp;attrId, bool val)</td>
<td>set bool value</td>
</tr>
<tr>
<td>bool GetBool (const Attr::BoolAttrId &amp;attrId) const</td>
<td>get bool value</td>
</tr>
<tr>
<td>void SetFloat (const Attr::FloatAttrId &amp;attrId, float val)</td>
<td>set float value</td>
</tr>
<tr>
<td>float GetFloat (const Attr::FloatAttrId &amp;attrId) const</td>
<td>get float value</td>
</tr>
<tr>
<td>void SetInt (const Attr::IntAttrId &amp;attrId, int val)</td>
<td>set int value</td>
</tr>
<tr>
<td>int GetInt (const Attr::IntAttrId &amp;attrId) const</td>
<td>get int value</td>
</tr>
<tr>
<td>void SetString (const Attr::StringAttrId &amp;attrId, const Util::String &amp;val)</td>
<td>set string value</td>
</tr>
<tr>
<td>GetString (const Attr::StringAttrId &amp;attrId)</td>
<td>get string value</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>const</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
</tr>
<tr>
<td>void SetFloat4 (const Attr::Float4AttrId &amp;attrId, const Math::float4 &amp;val)</td>
<td>set float4 value</td>
</tr>
<tr>
<td>Math::float4</td>
<td>GetFloat4 (const Attr::Float4AttrId &amp;attrId)</td>
</tr>
<tr>
<td>void</td>
<td>SetMatrix44 (const Attr::Matrix44AttrId &amp;attrId, const Math::matrix44 &amp;val)</td>
</tr>
<tr>
<td>const Math::matrix44 &amp;</td>
<td>GetMatrix44 (const Attr::Matrix44AttrId &amp;attrId)</td>
</tr>
<tr>
<td>void</td>
<td>SetGuid (const Attr::GuidAttrId &amp;attrId, const Util::Guid &amp;guid)</td>
</tr>
<tr>
<td>const Util::Guid &amp;</td>
<td>GetGuid (const Attr::GuidAttrId &amp;attrId)</td>
</tr>
<tr>
<td>void</td>
<td>SetBlob (const Attr::BlobAttrId &amp;attrId, const Util::Blob &amp;blob)</td>
</tr>
<tr>
<td>const Util::Blob &amp;</td>
<td>GetBlob (const Attr::BlobAttrId &amp;attrId)</td>
</tr>
<tr>
<td>void</td>
<td>SetAsyncEnabled (bool b)</td>
</tr>
<tr>
<td>bool</td>
<td>IsAsyncEnabled () const</td>
</tr>
<tr>
<td>void</td>
<td>SetResourceId (const ResourceId &amp;id)</td>
</tr>
<tr>
<td>const ResourceId &amp;</td>
<td>GetResourceId () const</td>
</tr>
<tr>
<td>void</td>
<td>SetLoader (const Ptr&lt; ResourceLoader &gt; &amp;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>Set optional resource loader</td>
</tr>
<tr>
<td><code>GetSaver()</code> const</td>
<td>Set optional resource saver</td>
</tr>
<tr>
<td><code>GetLoader()</code> const</td>
<td>Get optional resource loader</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> virtual State</td>
<td>Load the resource</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> virtual bool</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> void</td>
<td>Increment refcount by one</td>
</tr>
<tr>
<td><code>Release()</code> void</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 string</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</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>
</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

void SetState (State s)  
    set current state

void IncrUseCount ()  
    increment use count

void DecrUseCount ()  
    decrement use count
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.

void Models::Model::UnloadResources()

unload node resources

This method asks all model nodes to unload their resources.

Resource::State Models::Model::GetResourceState()

get the overall resource loading state (Initial -> Pending -> Loaded/Failed/Cancelled)

Checks all model nodes and returns a cumulative resource loading state.

Resource::State Resources::Resource::Load()

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.

```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::ModelInstance
Models::ModellInstance Class Reference

#include <modelinstance.h>

Inheritance diagram for Models::ModellInstance:

```
<table>
<thead>
<tr>
<th>Core::RefCounted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Models::ModellInstance</td>
</tr>
</tbody>
</table>
```
Detailed Description

A ModelInstance contains the per-instance data for rendering a Model. Usually there is one ModelInstance created per game object.

A ModelInstance is roughly comparable to a Nebula2 nRenderContext.

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

<table>
<thead>
<tr>
<th>Method</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>void <strong>Discard ()</strong></td>
<td>cleanup and unlink from model</td>
</tr>
<tr>
<td>bool <strong>IsAttachedToModel ()</strong> const</td>
<td>return true if currently attached to <em>Model</em></td>
</tr>
<tr>
<td>const <em>Ptr&lt; Model &gt; &amp;</em>* GetModel ()** const</td>
<td>get the <em>Model</em> this instance was created from</td>
</tr>
<tr>
<td>const <em>Ptr&lt; Graphics::ModelEntity &gt; &amp;</em>* GetModelEntity ()** const</td>
<td>get the <em>ModelEntity</em> 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 <em>Math::matrix44 &amp;</em>* GetTransform ()** const</td>
<td>get world space transform</td>
</tr>
<tr>
<td>void <strong>SetTime (Timing::Time t)</strong></td>
<td>set instance time</td>
</tr>
<tr>
<td><em>Timing::Time</em>* GetTime () const</td>
<td>get instance time</td>
</tr>
<tr>
<td>bool <strong>HasNodeInstance (const Util::Atom&lt; Util::String &gt; &amp;name)</strong> const</td>
<td>return true if a model instance node exists</td>
</tr>
<tr>
<td>const <em>Ptr&lt; ModelNodeInstance &gt; &amp;</em>* LookupNodeInstance (const Util::Atom&lt; Util::String &gt; &amp;name)** const</td>
<td>lookup an instance node</td>
</tr>
<tr>
<td>const <em>Util::Array&lt; <em>Ptr&lt; ModelNodeInstance &gt; &gt; &amp;</em></em> GetNodeInstances ()** const</td>
<td>get all instance nodes</td>
</tr>
<tr>
<td>const <em>Ptr&lt; ModelNodeInstance &gt; &amp;</em>* LookupNodeInstanceByRTTI (const Core::Rtti &amp;rtti)** const</td>
<td>lookup an instance node by rtti type</td>
</tr>
</tbody>
</table>
void Update ()
perform per-frame update (after setting transform, visibility, time, etc)

void RenderDebug ()
render node specific debug shape

void SetAllModelNodeInstancesVisible (bool b)
set all modelnode instances visible

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>
## Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>SetModelEntity</strong></td>
<td>(const Ptr&lt; Graphics::ModelEntity &gt; &amp;mdlEntity) set pointer to ModelEntity which owns this instance</td>
</tr>
<tr>
<td>void</td>
<td><strong>NotifyVisible</strong></td>
<td>(IndexT frameIndex) notify the model instance that it is currently visible</td>
</tr>
<tr>
<td>virtual</td>
<td><strong>OnAttachToModel</strong></td>
<td>(const Ptr&lt; Model &gt; &amp;model) called when attached to Model</td>
</tr>
<tr>
<td>virtual</td>
<td><strong>OnRemoveFromModel</strong></td>
<td>() called when removed from Model</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool Models::ModelInstance::HasNodeInstance(const Util::Atom<Util::String> & name) const

return true if a model instance node exists
```

Returns true if a ModelInstanceNode exists by name. ModelInstanceNodes are associated 1:1 with the ModelNodes of the Model which created this instance. Because of this we can just use the Model's node map.

```cpp
void Models::ModelInstance::Update()
```

perform per-frame update (after setting transform, visibility, time, etc)

The Update() method is called once per frame on potentially visible ModelInstances. The method must be called AFTER transformation, visibility, time or other parameters have been set. The Update() method will be propagated to all instance nodes.

```cpp
void Models::ModelInstance::RenderDebug()
```

render node specific debug shape

This method is called from the RenderDebug of Graphics::ModelEntity.

```cpp
void Models::ModelInstance::NotifyVisible(IndexT frameIndex) [protected]
```

notify the model instance that it is currently visible

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**.

```cpp
void Models::ModelInstance::OnAttachToModel(const Ptr<Model> m) [protected, virtual]
```
called when attached to **Model**

This method is called when this instance is attached to its **Model**. The most important thing happening at initialization is for each **ModelNode**, a corresponding ModelInstanceNode will be created, which holds ModelNode-per-instance data.

```cpp
void Models::ModelInstance::OnRemoveFromModel() [protected, virtual]
```
called when removed from **Model**

This method is called when this instance is removed from its **Model**.

```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.
Core::RefCounted::GetClassName

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::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
## Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ModelNode ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~ModelNode ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void <strong>ApplySharedState ()</strong></td>
<td>apply state shared by all my ModelNodeInstances</td>
</tr>
<tr>
<td>virtual <strong>GetResourceState ()</strong> const</td>
<td>get overall state of contained resources (Initial, Loaded, Pending, Failed, Cancelled)</td>
</tr>
<tr>
<td>const <strong>Util::Atom<a href="">Util::String</a> &amp;</strong> GetName() const</td>
<td>get model node name</td>
</tr>
<tr>
<td><strong>ModelNodeType::Code</strong> GetType() const</td>
<td>get the ModelNodeType</td>
</tr>
<tr>
<td>bool <strong>HasParent ()</strong> const</td>
<td>return true if node has a parent</td>
</tr>
<tr>
<td>const <strong>Ptr&lt;ModelNode&gt; &amp;</strong> GetParent() const</td>
<td>get parent node</td>
</tr>
<tr>
<td>const <strong>Util::Array&lt;Ptr&lt;ModelNode&gt;&gt; &amp;</strong> GetChildren() const</td>
<td>get child nodes</td>
</tr>
<tr>
<td>bool <strong>IsAttachedToModel ()</strong> const</td>
<td>return true if currently attached to a Model</td>
</tr>
<tr>
<td>const <strong>Ptr&lt;Model&gt; &amp;</strong> GetModel() const</td>
<td>get model this node is attached to</td>
</tr>
<tr>
<td>const <strong>Math::bbox &amp;</strong> GetBoundingBox() const</td>
<td>get bounding box of model node</td>
</tr>
<tr>
<td>const <strong>Attr::AttributeContainer &amp;</strong> GetAttrs() const</td>
<td>read access to model node attributes</td>
</tr>
<tr>
<td>bool <strong>HasAttr</strong> (const **Attr::AttrId &amp;attrId) const</td>
<td>check if model node attribute exists</td>
</tr>
<tr>
<td>void <strong>SetAttr</strong> (const <strong>Attr::Attribute &amp;attr)</strong></td>
<td>set generic model node attribute</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetAttr</code> (const <code>Attr::AttrId</code> &amp;attrId) const</td>
<td>get generic model node attribute</td>
</tr>
<tr>
<td><code>SetBool</code> (const <code>Attr::BoolAttrId</code> &amp;attrId, bool val)</td>
<td>set bool value</td>
</tr>
<tr>
<td><code>GetBool</code> (const <code>Attr::BoolAttrId</code> &amp;attrId) const</td>
<td>get bool value</td>
</tr>
<tr>
<td><code>SetFloat</code> (const <code>Attr::FloatAttrId</code> &amp;attrId, float val)</td>
<td>set float value</td>
</tr>
<tr>
<td><code>GetFloat</code> (const <code>Attr::FloatAttrId</code> &amp;attrId) const</td>
<td>get float value</td>
</tr>
<tr>
<td><code>SetInt</code> (const <code>Attr::IntAttrId</code> &amp;attrId, int val)</td>
<td>set int value</td>
</tr>
<tr>
<td><code>GetInt</code> (const <code>Attr::IntAttrId</code> &amp;attrId) const</td>
<td>get int value</td>
</tr>
<tr>
<td><code>SetString</code> (const <code>Attr::StringAttrId</code> &amp;attrId, const <code>Util::String</code> &amp;val)</td>
<td>set string value</td>
</tr>
<tr>
<td><code>GetString</code> (const <code>Attr::StringAttrId</code> &amp;attrId) const</td>
<td>get string value</td>
</tr>
<tr>
<td><code>SetFloat4</code> (const <code>Attr::Float4AttrId</code> &amp;attrId, const <code>Math::float4</code> &amp;val)</td>
<td>set float4 value</td>
</tr>
<tr>
<td><code>GetFloat4</code> (const <code>Attr::Float4AttrId</code> &amp;attrId) const</td>
<td>get float4 value</td>
</tr>
<tr>
<td><code>SetMatrix44</code> (const <code>Attr::Matrix44AttrId</code> &amp;attrId, const <code>Math::matrix44</code> &amp;val)</td>
<td>set matrix44 value</td>
</tr>
<tr>
<td><code>GetMatrix44</code> (const <code>Attr::Matrix44AttrId</code> &amp;attrId) const</td>
<td>get matrix44 value</td>
</tr>
<tr>
<td><code>SetGuid</code> (const <code>Attr::GuidAttrId</code> &amp;attrId,</td>
<td>set guid value</td>
</tr>
<tr>
<td><code>Math::matrix44</code> &amp;</td>
<td></td>
</tr>
<tr>
<td><code>Math::float4</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>void</td>
<td>set guid value</td>
</tr>
<tr>
<td>const Util::Guid &amp;</td>
<td>GetGuid (const Attr::GuidAttrId &amp;attrId) const</td>
</tr>
<tr>
<td>void</td>
<td>set blob value</td>
</tr>
<tr>
<td>const Util::Blob &amp;</td>
<td>GetBlob (const Attr::BlobAttrId &amp;attrId) const</td>
</tr>
<tr>
<td>void</td>
<td>set model node name</td>
</tr>
<tr>
<td>void</td>
<td>set ModelNodeType</td>
</tr>
<tr>
<td>void</td>
<td>set parent node</td>
</tr>
<tr>
<td>void</td>
<td>add a child node</td>
</tr>
<tr>
<td>virtual void</td>
<td>LoadFromAttrs (const Attr::AttributeContainer &amp;attrs) called to initialize from attributes</td>
</tr>
<tr>
<td>void</td>
<td>AddVisibleNodeInstance (IndexT frameIndex, const Ptr&lt; ModelNodeInstance &gt; &amp;nodeInst) called by model node instance on NotifyVisible()</td>
</tr>
<tr>
<td>int</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void</td>
<td>increment refcount by one</td>
</tr>
<tr>
<td>void</td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td>return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool</td>
<td>IsInstanceOf (const Util::String) const</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 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 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 SetBoundingBox (const Math::bbox &amp;b)</code></td>
<td>set bounding box</td>
</tr>
<tr>
<td>virtual <code>Ptr &lt; ModelNodeInstance &gt;</code> CreateNodeInstance () const</td>
<td>create a model node instance</td>
</tr>
<tr>
<td>virtual void OnAttachToModel (const Ptr &lt; Model &gt; &amp;model)</td>
<td>called when attached to model node</td>
</tr>
<tr>
<td>virtual void OnRemoveFromModel ()</td>
<td>called when removed from model node</td>
</tr>
<tr>
<td>virtual void SaveToAttrs (Attr::AttributeContainer &amp;attrs)</td>
<td>called to save state back to attributes</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>const Util::Array &lt; Ptr &lt; ModelNodeInstance &gt; &gt; &amp; GetVisibleModelNodeInstances (ModelNodeType::Code t) const</td>
<td>get visible model node instances</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void Models::ModelNode::ApplySharedState() [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 `Models::ShapeNode`, `Models::SkinShapeNode`, 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 `Models::CharacterNode`, `Models::ShapeNode`, and `Models::StateNode`.

```cpp
void Models::ModelNode::LoadResources() [protected, virtual]
```

called when resources should be loaded

This method is called when required resources should be loaded.
Reimplemented in Models::CharacterNode, Models::ParticleSystemNode, Models::ShapeNode, Models::SkinShapeNode, and Models::StateNode.

```cpp
void Models::ModelNode::UnloadResources() [protected, virtual]
```
called when resources should be unloaded

This method is called when required resources should be unloaded.

Reimplemented in Models::CharacterNode, Models::ParticleSystemNode, Models::ShapeNode, and Models::StateNode.

```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::ModelNodeInstance
Models::ModelNodeInstance Class Reference

#include <modelnodeinstance.h>

Inheritance diagram for Models::ModelNodeInstance:

```
Core::RefCounted

Models::ModelNodeInstance

Models::TransformNodeInstance

Models::CharacterNodeInstance  Models::StateNodeInstance

Models::ShapeNodeInstance

Models::ParticleSystemNodeInstance  Models::SkinShapeNodeInstance
```
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>Update ()</strong></td>
<td>perform per-frame updates</td>
</tr>
<tr>
<td>virtual void <strong>ApplyState ()</strong></td>
<td>apply per-instance state prior to rendering</td>
</tr>
<tr>
<td>virtual void <strong>Render ()</strong></td>
<td>perform rendering</td>
</tr>
<tr>
<td>bool <strong>HasParent ()</strong> const</td>
<td>return true if node has a parent</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; ModelNodeInstance &gt; &amp;</strong> <strong>GetParent ()</strong> const</td>
<td>get parent node</td>
</tr>
<tr>
<td>const <strong>Util::Array&lt; Ptr&lt; ModelNodeInstance &gt; &gt; &amp;</strong> <strong>GetChildren ()</strong> const</td>
<td>get child nodes</td>
</tr>
<tr>
<td>bool <strong>IsAttachedToModellInstance ()</strong> const</td>
<td>return true if attached to <strong>ModellInstance</strong></td>
</tr>
<tr>
<td>const <strong>Ptr&lt; ModellInstance &gt; &amp;</strong> <strong>GetModellInstance ()</strong> const</td>
<td>get the <strong>ModellInstance</strong> we are attached to</td>
</tr>
<tr>
<td>const <strong>Ptr&lt; ModelNode &gt; &amp;</strong> <strong>GetModelNode ()</strong> const</td>
<td>get the <strong>ModelNode</strong> we're associated with</td>
</tr>
<tr>
<td>virtual void <strong>SetVisible</strong> (bool b)</td>
<td>set visible, used by charactersystem</td>
</tr>
<tr>
<td>bool <strong>IsVisible ()</strong> const</td>
<td>is visible</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>
</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</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 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>
<tr>
<td><code>virtual void OnAttachToModelInstance (const Ptr&lt; ModelInstance &gt; &amp;inst, constPtr&lt; ModelNode &gt; &amp;node, constPtr&lt; ModelNodeInstance &gt; &amp;parentNodeInst)</code></td>
<td>called when attached to <code>ModelInstance</code></td>
</tr>
<tr>
<td><code>virtual void OnRemoveFromModelInstance ()</code></td>
<td>called when removed from <code>ModelInstance</code></td>
</tr>
<tr>
<td><code>virtual void OnNotifyVisible (IndexT frameIndex)</code></td>
<td>notify that we are visible</td>
</tr>
<tr>
<td><code>virtual void RenderDebug ()</code></td>
<td>render node specific debug shape</td>
</tr>
<tr>
<td><code>void SetChildrenVisibility (ModelNodeInstance *parent, bool b)</code></td>
<td>set visible flag of children</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::ModelNodeInstance::Update() [virtual]
```

perform per-frame updates

The `Update()` method is called exactly once per frame before visibility culling on every *potentially* visible model node instance. Actions performed in the `Update()` usually include resolving the world space transformation (if changed) and performing animation. The `Update()` method will be called from `ModelInstance::Update()` in "hierarchy order" (parent nodes will called before their child nodes).

Reimplemented in `Models::CharacterNodeInstance`, `Models::ParticleSystemNodeInstance`, `Models::SkinShapeNodeInstance`, and `Models::TransformNodeInstance`.

```cpp
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`, and `Models::TransformNodeInstance`.

```cpp
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()` invocation.

Reimplemented in `Models::ParticleSystemNodeInstance`, `Models::ShapeNodeInstance`, and `Models::SkinShapeNodeInstance`.

```cpp
void Models::ModelNodeInstance::SetVisible(bool b) [inline, virtual]
```

set visible, used by `charactersystem`

Set visibility of node and its children, should not be call per frame!

Reimplemented in `Models::SkinShapeNodeInstance`.

```cpp
void Models::ModelNodeInstance::OnNotifyVisible(IndexT frameIndex) [protected, virtual]
```

notify that we are visible

This method is called from the `NotifyVisible()` method of our `ModelInstance` object. If the `ModelNodeInstance` provides something renderable it should respond by adding itself as visible node instance to its model node.

Reimplemented in `Models::ParticleSystemNodeInstance`, `Models::ShapeNodeInstance`, and `Models::SkinShapeNodeInstance`.

```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()
```

[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.
Models::ModelNodeType
Models::ModelNodeType Class Reference

#include <modelnodetype.h>
Detailed Description

ModelNodeTypes identify a ModelNode for a specific rendering pass.

(C) 2007 Radon Labs GmbH
## Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>type enum</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 Tue Feb 19 12:16:51 2008
Models::ModelReader
Models::ModelReader Class Reference

#include <modelreader.h>

Inheritance diagram for Models::ModelReader:
Detailed Description

Implements a stream reader for Model persistency. This is just a base class for more specialized stream readers which implement different file formats (binary, XML, etc...).

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual ~ModelReader ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual bool Open ()=0</td>
<td>Begin reading from the stream</td>
</tr>
<tr>
<td>virtual void Close ()=0</td>
<td>End reading from the stream</td>
</tr>
<tr>
<td>void SetModelResId (const Resources::ResourceId &amp;resId)</td>
<td>Set the read cursor to the first Model in the stream</td>
</tr>
<tr>
<td>const Resources::ResourceId &amp; GetModelResId () const</td>
<td>Get the model resource id</td>
</tr>
<tr>
<td>void SetModel (const Ptr&lt;Model&gt; &amp;model)</td>
<td>Get attributes of current model</td>
</tr>
<tr>
<td>virtual bool FillModel ()=0</td>
<td>Parse resource and build model hierarchy</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>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</td>
<td><code>IsInstanceOf (const Util::String &amp;className)</code> 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 (const Util::FourCC &amp;classFourCC)</code> 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 (const Rtti &amp;rtti)</code> 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 (const Util::String &amp;rttiName)</code> 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 (const Util::FourCC &amp;rttiFourCC)</code> 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><code>GetClassName ()</code> 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><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

static void **DumpRefCountingLeaks**( )

*dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)*
void Models::ModelReader::SetModelResId(
    const Resources::ResourceId& resId)

set the read cursor to the first Model in the stream

set the read cursor to the next Model in the stream set model resource id (some file format's don't provide their own)

void Models::ModelReader::SetModel(
    const Ptr<Model>& model)[inline]

get attributes of current model

set the read cursor to the first ModelNode in the current Model set the read cursor to the next ModelNode in the current Model get the model node name get model node class get the name of the parent node get attributes of current model node set model for filling up with modelnodes and attributes

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.

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.
```
Models::ModelServer
Models::ModelServer Class Reference

#include <modelserver.h>

Inheritance diagram for Models::ModelServer:
Detailed Description

The **ModelServer** loads and creates shared **Model** objects.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</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 Ptr&lt; Resources::ResourceMapper &gt;&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_PTR&lt; ManagedModel &gt;&amp;managedModel)</strong></td>
<td>discard a managed model</td>
</tr>
<tr>
<td><strong>SaveModel (const_PTR&lt; Model &gt;&amp;model, const IO::URI &amp;uri, const Core::Rtti &amp;saverClass)</strong></td>
<td>save a Model to URI using the provided model</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</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><strong>bool IsInstanceOf(const Rtti &amp;rtti)</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)</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)</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)</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()</strong></td>
<td>Get the class name</td>
</tr>
<tr>
<td><strong>const Util::FourCC GetClassFourCC()</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 (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.
Models::ModelWriter
#include <modelwriter.h>

Inheritance diagram for Models::ModelWriter:

Core::RefCounted

IO::StreamWriter

Models::ModelWriter

Models::BinaryModelWriter  Models::XmlModelWriter
**Detailed Description**

Implements a stream writer class for Model persistency. This is just a base class for more specialized stream writers which implement different file formats.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>~ModelWriter ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>virtual Util::String GetFileExtension () const =0</code></td>
<td>get the file extension used by the writer</td>
</tr>
<tr>
<td><code>virtual bool Open ()=0</code></td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td><code>virtual void Close ()=0</code></td>
<td>end reading from the stream</td>
</tr>
<tr>
<td><code>virtual bool BeginModel (const Ptr&lt; Model &gt; &amp;model)=0</code></td>
<td>begin writing a new Model</td>
</tr>
<tr>
<td><code>virtual void WriteModelAttrs (const Attr::AttributeContainer &amp;attrs)=0</code></td>
<td>write model attributes</td>
</tr>
<tr>
<td><code>virtual void EndModel ()=0</code></td>
<td>end writing current Model</td>
</tr>
<tr>
<td><code>virtual bool BeginModelNode (const Ptr&lt; ModelNode &gt; &amp;modelNode)=0</code></td>
<td>begin writing a new ModelNode</td>
</tr>
<tr>
<td><code>virtual void WriteModelNodeAttrs (const Attr::AttributeContainer &amp;attrs)=0</code></td>
<td>write mode node attributes</td>
</tr>
<tr>
<td><code>virtual void EndModelNode ()=0</code></td>
<td>end writing current ModelNode</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 () 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 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>
</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 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::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.

```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::N2ModelReader
Models::N2ModelReader Class Reference

#include <n2modelreader.h>

Inheritance diagram for Models::N2ModelReader:
Detailed Description

**Legacy** N2 binary reader, reads a subset of .n2 files into a **Model**. NOTE: This class is very inefficient and shouldn't be used for production code.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>N2ModelReader()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~N2ModelReader()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>Open()</code></td>
<td>begin reading from the stream</td>
</tr>
<tr>
<td><code>Close()</code></td>
<td>end reading from the stream</td>
</tr>
<tr>
<td><code>FillModel()</code></td>
<td>parse resource and build model hierarchy</td>
</tr>
<tr>
<td><code>SetModelResId()</code></td>
<td>set the read cursor to the first <code>Model</code> in the stream</td>
</tr>
<tr>
<td><code>GetModelResId()</code></td>
<td>get the model resource id</td>
</tr>
<tr>
<td><code>SetModel()</code></td>
<td>get attributes of current model</td>
</tr>
<tr>
<td><code>SetStream()</code></td>
<td>set stream to read from</td>
</tr>
<tr>
<td><code>GetStream()</code></td>
<td>get currently set stream</td>
</tr>
<tr>
<td><code>HasStream()</code></td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td><code>Eof()</code></td>
<td>return true if the stream has reached EOF</td>
</tr>
<tr>
<td><code>IsOpen()</code></td>
<td>return true if currently open</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>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

```
void Models::ModelReader::SetModelResId(const Resources::ResourceId& resId) [inherited]
```  
set the read cursor to the first `Model` in the stream
set the read cursor to the next `Model` in the stream set model resource id (some file format's don't provide their own)

```
void Models::ModelReader::SetModel(const Ptr<Model>& model) [inline, inherited]
```  
get attributes of current model
set the read cursor to the first `ModelNode` in the current `Model` set the read cursor to the next `ModelNode` in the current `Model` get the model node name get model node class get the name of the parent node get attributes of current model node set model for filling up with modelnodes and attributes

```
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`.

```
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.
```
Models::ParticleSystemNode
Models::ParticleSystemNode Class Reference

#include <particlesystemnode.h>

Inheritance diagram for Models::ParticleSystemNode:
Detailed Description

A model node which holds particle system information and applies a shared dynamicMeshes for its set of ParticleSystemNodeInstances.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ParticleSystemNode ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~ParticleSystemNode ()</strong></td>
<td>destructor</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><strong>Ptr<a href="">ParticleSystem::DynamicMesh</a> GetParticleMesh () const</strong></td>
<td>get the particle mesh</td>
</tr>
<tr>
<td>void <strong>SetInvisible</strong> (bool value)</td>
<td>set if invisible or not</td>
</tr>
<tr>
<td>void <strong>SetEmissionDuration</strong> (float time)</td>
<td>set the end time</td>
</tr>
<tr>
<td>float <strong>GetEmissionDuration</strong> () const</td>
<td>get the emission duration</td>
</tr>
<tr>
<td>void <strong>SetLoop</strong> (bool b)</td>
<td>set if loop emitter or not</td>
</tr>
<tr>
<td>bool <strong>GetLoop</strong> () const</td>
<td>is loop emitter ?</td>
</tr>
<tr>
<td>void <strong>SetActivityDistance</strong> (float activityDistance)</td>
<td>set the activity distance</td>
</tr>
<tr>
<td>float <strong>GetActivityDistance</strong> () const</td>
<td>get the activity distance</td>
</tr>
<tr>
<td>void <strong>SetStartRotationMin</strong> (float rotMin)</td>
<td>set the maximum start rotation angle</td>
</tr>
<tr>
<td>void <strong>SetStartRotationMax</strong> (float rotMax)</td>
<td>set the maximum start rotation angle</td>
</tr>
<tr>
<td>void <strong>SetGravity</strong> (float gravity)</td>
<td>set gravity</td>
</tr>
<tr>
<td>float <strong>GetGravity</strong> () const</td>
<td>get gravity</td>
</tr>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
void SetParticleStretch (float particleStretch)
set amount of stretching

void SetParticleVelocityRandomize (float velRandom)
set velocity randomize value

void SetParticleRotationRandomize (float rotRandom)
set rotation randomize value

void SetParticleSizeRandomize (float sizeRandom)
set size randomize value

void SetRandomRotDir (bool isRandom)
randomize rotation direction on/off

void SetTileTexture (int tileTexture)
set texture tiling parts

void SetStretchToStart (bool isStretchToStart)
set if texture should be stretched to the emission start point

void SetPrecalcTime (float preClacTime)
set precalc time

void SetRenderOldestFirst (bool isOldestFirst)
set wether to render oldest or youngest particles first

void SetStretchDetail (int stretchDetail)
set the stretch detail

void SetViewAngleFade (bool viewAngelFade)
set wether to render oldest or youngest particles first

bool GetRenderOldestFirst () const
get wether to render oldest or youngest particles first

void SetStartDelay (float startDelay)
set start delay

void SetCurve (ParticleSystem::ParticleEmitter::CurveType curveType, const ParticleSystem::EnvelopeCurve &curve)
set one of the envelope curves (not the color)

const ParticleSystem::EnvelopeCurve & GetCurve (ParticleSystem::ParticleEmitter::CurveType curveType)
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetRGBCurve(const ParticleSystem::Vector3EnvelopeCurve &amp;curve)</code></td>
<td>set the particle rgb curve</td>
</tr>
<tr>
<td><code>const ParticleSystem::Vector3EnvelopeCurve &amp; GetRGBCurve()</code> const</td>
<td>get the particle rgb curve</td>
</tr>
<tr>
<td><code>void SetBillboardOrientation(bool b)</code></td>
<td>sets orientation to billboard</td>
</tr>
<tr>
<td><code>bool GetBillboardOrientation()</code></td>
<td>gets billboard flag</td>
</tr>
<tr>
<td><code>virtual Resources::Resource::State GetResourceState()</code> const</td>
<td>get overall state of contained resources (Initial, Loading, Pending, Failed, Cancelled)</td>
</tr>
<tr>
<td><code>virtual void ApplySharedState()</code></td>
<td>apply state shared by all my ModelNodeInstances</td>
</tr>
<tr>
<td><code>const Ptr&lt;Resources::ManagedMesh&gt; &amp; GetManagedMesh()</code> const</td>
<td>get managed mesh</td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::ShaderInstance&gt; &amp; GetShaderInstance()</code> const</td>
<td>get pointer to contained shader instance</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> const</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> const</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> const</td>
<td>get scale</td>
</tr>
<tr>
<td><code>void SetRotatePivot(const Math::point &amp;p)</code></td>
<td>set rotate pivot</td>
</tr>
</tbody>
</table>
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

const Util::Atom < Util::String > & **GetName** () const
get model node name

ModelNodeType::Code **GetType** () const
get the ModelNodeType

bool **HasParent** () const
return true if node has a parent

const Ptr< ModelNode > & **GetParent** () const
get parent node

const Util::Array < Ptr< ModelNode > > & **GetChildren** () const
get child nodes

bool **IsAttachedToModel** () const
return true if currently attached to a Model

const Ptr< Model > & **GetModel** () const
get model this node is attached to

const Math::bbox & **GetBoundingBox** () const
get bounding box of model node

const Attr::AttributeContainer & **GetAttrs** () const
read access to model node attributes

bool **HasAttr** (const Attr::AttrId &attrId) const
check if model node attribute exists

void **SetAttr** (const Attr::Attribute &attr)
set generic model node attribute

const Attr::Attribute & **GetAttr** (const Attr::AttrId &attrId) const
get generic model node attribute

void **SetBool** (const Attr::BoolAttrId &attrId, bool val)
set bool value

bool **GetBool** (const Attr::BoolAttrId &attrId) const
void SetFloat (const Attr::FloatAttrId &attrId, float val)  
set float value

float GetFloat (const Attr::FloatAttrId &attrId) const  
get float value

void SetInt (const Attr::IntAttrId &attrId, int val)  
set int value

int GetInt (const Attr::IntAttrId &attrId) const  
get int value

void SetString (const Attr::StringAttrId &attrId, const Util::String &val)  
set string value

const Util::String & GetString (const Attr::StringAttrId &attrId) const  
get string value

void SetFloat4 (const Attr::Float4AttrId &attrId, const Math::float4 &val)  
set float4 value

Math::float4 GetFloat4 (const Attr::Float4AttrId &attrId) const  
get float4 value

void SetMatrix44 (const Attr::Matrix44AttrId &attrId, const Math::matrix44 &val)  
set matrix44 value

const Math::matrix44 & GetMatrix44 (const Attr::Matrix44AttrId &attrId) const  
get matrix44 value

void SetGuid (const Attr::GuidAttrId &attrId, const Util::Guid &guid)  
set guid value

const Util::Guid & GetGuid (const Attr::GuidAttrId &attrId) const  
get guid value

void SetBlob (const Attr::BlobAttrId &attrId, const Util::Blob &blob)
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const Util::Blob &amp; GetBlob (const Attr::BlobAttrId &amp;attrId)</code></td>
<td>Get blob value</td>
</tr>
<tr>
<td><code>void SetName (const Util::Atom&lt; Util::String &amp;n)</code></td>
<td>Set model node name</td>
</tr>
<tr>
<td><code>void SetType (ModelNodeType::Code t)</code></td>
<td>Set <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>void AddChild (const Ptr&lt;ModelNode&gt; &amp;)</code></td>
<td>Add a child node</td>
</tr>
<tr>
<td><code>void 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>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><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>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

<table>
<thead>
<tr>
<th>static void DumpRefCountingLeaks ()</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dump refcounting leaks, call at end of application (NEBUŁA3_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 \texttt{Ptr} \texttt{&lt; ModelNodeInstance &gt;} \texttt{CreateNodeInstance} () const</td>
<td>gets the particle system mesh;</td>
</tr>
<tr>
<td>void \texttt{SetupManagedTextureVariable} (const \texttt{Attr::AttrId} &amp;resAttrId, const \texttt{Ptr&lt; CoreGraphics::ShaderVariable&gt;} &amp;ivar)</td>
<td>setup a new managed texture variable</td>
</tr>
<tr>
<td>void \texttt{UpdateManagedTextureVariables} ()</td>
<td>update managed texture variables</td>
</tr>
<tr>
<td>virtual void \texttt{LoadFromAttrs} (const \texttt{Attr::AttributeContainer} &amp;attrs)</td>
<td>called to initialize from attributes</td>
</tr>
<tr>
<td>virtual void \texttt{SaveToAttrs} (\texttt{Attr::AttributeContainer} &amp;attrs)</td>
<td>called to save state back to attributes</td>
</tr>
<tr>
<td>void \texttt{SetBoundingBox} (const \texttt{Math::bbox} &amp;b)</td>
<td>set bounding box</td>
</tr>
<tr>
<td>virtual void \texttt{OnAttachToModel} (const \texttt{Ptr&lt; Model&gt;} &amp;model)</td>
<td>called when attached to model node</td>
</tr>
<tr>
<td>virtual void \texttt{OnRemoveFromModel} ()</td>
<td>called when removed from model node</td>
</tr>
<tr>
<td>const \texttt{Util::Array} \texttt{&lt; Ptr&lt; ModelNodeInstance &gt;&gt;} &amp; \texttt{GetVisibleModelNodeInstances} (ModelNodeType::Code t) const</td>
<td>get visible model node instances</td>
</tr>
</tbody>
</table>
## Protected Attributes

<table>
<thead>
<tr>
<th>float</th>
<th><code>sampledCurves</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ParticleSystem::ParticleEmitter::ParticleTimeDetail]</td>
</tr>
<tr>
<td></td>
<td>[ParticleSystem::ParticleEmitter::CurveTypeCount]</td>
</tr>
</tbody>
</table>

A mesh for storing and rendering dynamic data. In this case particle data.
Member Function Documentation

`Ptr< ModelNodeInstance >` Models::ParticleSystemNode::CreateNodeInstance ( ) const [protected, virtual]

gets the particle system mesh;
create a model node instance
Reimplemented from `Models::ShapeNode`.

void Models::StateNode::SetupManagedTextureVariable ( const Attr::AttrId & resAttrId,
  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.

void Models::StateNode::UpdateManagedTextureVariables ( ) [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.

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::ParticleSystemNodeInstance
Models::ParticleSystemNodeInstance Class Reference

#include <particlesystemnodeinstance.h>

Inheritance diagram for Models::ParticleSystemNodeInstance:

```
Core::RefCounted
  ↓
Models::ModelNodeInstance
  ↓
Models::TransformNodeInstance
  ↓
Models::StateNodeInstance
  ↓
Models::ShapeNodeInstance
  ↓
Models::ParticleSystemNodeInstance
```
Detailed Description

Holds a particle system emitter and applies it.

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

<table>
<thead>
<tr>
<th>Function</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 Update ()</code></td>
<td>Perform per-frame updates</td>
</tr>
<tr>
<td><code>virtual void Render ()</code></td>
<td>Perform rendering</td>
</tr>
<tr>
<td><code>void SetEmitter (Ptr&lt;ParticleSystem::ParticleEmitter&gt; e)</code></td>
<td>Sets the emitter which is used by the instance</td>
</tr>
<tr>
<td><code>virtual void ApplyState ()</code></td>
<td>Apply per-instance state prior to rendering</td>
</tr>
<tr>
<td><code>Ptr&lt;CoreGraphics::ShaderVariableInstance&gt; CreateShaderVariableInstance (const CoreGraphics::ShaderVariable::Semantic &amp;semantic)</code></td>
<td>Instantiate a shader variable by semantic</td>
</tr>
<tr>
<td><code>bool HasShaderVariableInstance (const CoreGraphics::ShaderVariable::Semantic &amp;semantic)</code> const</td>
<td>Return true if a shader variable has been instantiated</td>
</tr>
<tr>
<td><code>const Ptr&lt;CoreGraphics::ShaderVariableInstance&gt; &amp; GetShaderVariableInstance (const CoreGraphics::ShaderVariable::Semantic &amp;semantic)</code> const</td>
<td>Get a shader variable instance</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 SetRotate (const Math::quaternion &amp;q)</code></td>
<td>Set rotate quaternion</td>
</tr>
<tr>
<td><code>const Math::quaternion &amp; GetRotate () const</code></td>
<td>Get rotate quaternion</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>void SetScale (const Math::vector &amp;s)</td>
<td>set scale</td>
</tr>
<tr>
<td>const Math::vector &amp; GetScale () const</td>
<td>get scale</td>
</tr>
<tr>
<td>void SetRotatePivot (const Math::point &amp;)</td>
<td>set rotate pivot</td>
</tr>
<tr>
<td>const Math::point &amp; GetRotatePivot () const</td>
<td>get rotate pivot</td>
</tr>
<tr>
<td>void SetScalePivot (const Math::point &amp;)</td>
<td>set scale pivot</td>
</tr>
<tr>
<td>const Math::point &amp; GetScalePivot () const</td>
<td>get scale pivot</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; GetLocalTransform ()</td>
<td>get resulting local transform matrix in local parent space</td>
</tr>
<tr>
<td>const Math::matrix44 &amp; GetModelTransform () const</td>
<td>get model space transform (valid after Update())</td>
</tr>
<tr>
<td>bool HasParent () const</td>
<td>return true if node has a parent</td>
</tr>
<tr>
<td>const Ptr&lt; ModelNodeInstance &gt; &amp; GetParent () const</td>
<td>get parent node</td>
</tr>
<tr>
<td>const Util::Array&lt; Ptr&lt; ModelNodeInstance &gt; &gt; &amp; GetChildren () const</td>
<td>get child nodes</td>
</tr>
<tr>
<td>bool IsAttachedToModelInstance () const</td>
<td>return true if attached to ModelInstance</td>
</tr>
<tr>
<td>const Ptr&lt; ModelInstance &gt; &amp; GetModelInstance () const</td>
<td>get the ModelInstance we are attached to</td>
</tr>
<tr>
<td>const Ptr&lt; ModelNode &gt; &amp; GetModelNode () const</td>
<td>get the ModelNode we’re associated with</td>
</tr>
<tr>
<td>virtual void SetVisible (bool b)</td>
<td>set visible, used by character system</td>
</tr>
<tr>
<td>bool IsVisible () const</td>
<td>is visible</td>
</tr>
<tr>
<td>int GetRefCount () const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>void AddRef ()</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>increment refcount by one</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void Release ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>bool 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>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool 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>
</tr>
</tbody>
</table>

<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 derived class</td>
</tr>
</tbody>
</table>

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

<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 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 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 OnAttachToModelInstance (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>virtual void OnRemoveFromModelInstance ()</code></td>
<td>Called when removed from <code>ModelInstance</code></td>
</tr>
<tr>
<td><code>virtual void OnNotifyVisible (IndexT frameIndex)</code></td>
<td>Notify that we are visible</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>
<tr>
<td><code>virtual void RenderDebug ()</code></td>
<td>Render node specific debug shape</td>
</tr>
<tr>
<td><code>void SetChildrenVisibility (ModelNodeInstance *parent, bool)</code></td>
<td>Set visible flag of children</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::ModelNodeInstance::SetVisible (bool b ) [inline, virtual, inherited]
```
set visible, used by charactersystem

Set visibility of node and its children, should not be call per frame!

Reimplemented in `Models::SkinShapeNodeInstance`.

```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::ShapeNode
#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
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShapeNode ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~ShapeNode ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>virtual Resources::Resource::State</strong></td>
<td>GetResourceState () const</td>
</tr>
<tr>
<td></td>
<td>get overall state of contained resources (Initial, Loaded, Pending, Failed, Cancelled)</td>
</tr>
<tr>
<td><strong>virtual void ApplySharedState ()</strong></td>
<td>apply state shared by all my ModelNodeInstances</td>
</tr>
<tr>
<td><strong>const Ptr &lt; Resources::ManagedMesh &gt; &amp;</strong></td>
<td>GetManagedMesh () const</td>
</tr>
<tr>
<td></td>
<td>get managed mesh</td>
</tr>
<tr>
<td><strong>const Ptr &lt; CoreGraphics::ShaderInstance &gt; &amp;</strong></td>
<td>GetShaderInstance () const</td>
</tr>
<tr>
<td></td>
<td>get pointer to contained shader instance</td>
</tr>
<tr>
<td><strong>void SetPosition (const Math::point &amp;p)</strong></td>
<td>SetPosition () const</td>
</tr>
<tr>
<td></td>
<td>set position</td>
</tr>
<tr>
<td><strong>const Math::point &amp;</strong></td>
<td>GetPosition () const</td>
</tr>
<tr>
<td></td>
<td>get position</td>
</tr>
<tr>
<td><strong>void SetRotate (const Math::quaternion &amp;r)</strong></td>
<td>SetRotate () const</td>
</tr>
<tr>
<td></td>
<td>set rotate quaternion</td>
</tr>
<tr>
<td><strong>const Math::quaternion &amp;</strong></td>
<td>GetRotate () const</td>
</tr>
<tr>
<td></td>
<td>get rotate quaternion</td>
</tr>
<tr>
<td><strong>void SetScale (const Math::vector &amp;s)</strong></td>
<td>SetScale () const</td>
</tr>
<tr>
<td></td>
<td>set scale</td>
</tr>
<tr>
<td><strong>const Math::vector &amp;</strong></td>
<td>GetScale () const</td>
</tr>
<tr>
<td></td>
<td>get scale</td>
</tr>
<tr>
<td><strong>void SetRotatePivot (const Math::point &amp;p)</strong></td>
<td>SetRotatePivot () const</td>
</tr>
<tr>
<td></td>
<td>set rotate pivot</td>
</tr>
<tr>
<td><strong>const Math::point &amp;</strong></td>
<td>GetRotatePivot () const</td>
</tr>
<tr>
<td></td>
<td>get rotate pivot</td>
</tr>
<tr>
<td><strong>SetScalePivot (const Math::point</strong></td>
<td>SetScalePivot ()</td>
</tr>
<tr>
<td></td>
<td>set scale</td>
</tr>
</tbody>
</table>
void &p)

set scale pivot

const Math::point & GetScalePivot () const
get scale pivot

const Util::Atom < Util::String > & GetName () const
get model node name

ModelNodeType::Code GetType () const
get the ModelNodeType

bool HasParent () const
return true if node has a parent

const Ptr< ModelNode > & GetParent () const
get parent node

const Util::Array < Ptr< ModelNode > > & GetChildren () const
get child nodes

bool IsAttachedToModel () const
return true if currently attached to a Model

const Ptr< Model > & GetModel () const
get model this node is attached to

const Math::bbox & GetBoundingBox () const
get bounding box of model node

const Attr::AttributeContainer & GetAttrs () const
read access to model node attributes

bool HasAttr (const Attr::AttrId &attrId) const
check if model node attribute exists

void SetAttr (const Attr::Attribute &attr)
set generic model node attribute

const Attr::Attribute & GetAttr (const Attr::AttrId &attrId) const
get generic model node attribute

void SetBool (const Attr::BoolAttrId &attrId, bool val)
set bool value

bool GetBool (const Attr::BoolAttrId &attrId) const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetFloat (const Attr::FloatAttrId &amp;attrId, float val)</code></td>
<td><code>get bool value</code></td>
<td><code>set float value</code></td>
</tr>
<tr>
<td><code>float GetFloat (const Attr::FloatAttrId &amp;attrId)</code></td>
<td><code>set float value</code></td>
<td><code>get float value</code></td>
</tr>
<tr>
<td><code>void SetInt (const Attr::IntAttrId &amp;attrId, int val)</code></td>
<td><code>set int value</code></td>
<td><code>get int value</code></td>
</tr>
<tr>
<td><code>int GetInt (const Attr::IntAttrId &amp;attrId)</code></td>
<td><code>set int value</code></td>
<td><code>get int value</code></td>
</tr>
<tr>
<td><code>void SetString (const Attr::StringAttrId &amp;attrId, const Util::String &amp;val)</code></td>
<td><code>set string value</code></td>
<td><code>get string value</code></td>
</tr>
<tr>
<td><code>const Util::String &amp; GetString (const Attr::StringAttrId &amp;attrId)</code></td>
<td><code>set string value</code></td>
<td><code>get string value</code></td>
</tr>
<tr>
<td><code>void SetFloat4 (const Attr::Float4AttrId &amp;attrId, const Math::float4 &amp;val)</code></td>
<td><code>set float4 value</code></td>
<td><code>get float4 value</code></td>
</tr>
<tr>
<td><code>Math::float4 GetFloat4 (const Attr::Float4AttrId &amp;attrId)</code></td>
<td><code>set float4 value</code></td>
<td><code>get float4 value</code></td>
</tr>
<tr>
<td><code>void SetMatrix44 (const Attr::Matrix44AttrId &amp;attrId, const Math::matrix44 &amp;val)</code></td>
<td><code>set matrix44 value</code></td>
<td><code>get matrix44 value</code></td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetMatrix44 (const Attr::Matrix44AttrId &amp;attrId)</code></td>
<td><code>set matrix44 value</code></td>
<td><code>get matrix44 value</code></td>
</tr>
<tr>
<td><code>void SetGuid (const Attr::GuidAttrId &amp;attrId, const Util::Guid &amp;guid)</code></td>
<td><code>set guid value</code></td>
<td><code>get guid value</code></td>
</tr>
<tr>
<td><code>const Util::Guid &amp; GetGuid (const Attr::GuidAttrId &amp;attrId)</code></td>
<td><code>set guid value</code></td>
<td><code>get guid value</code></td>
</tr>
</tbody>
</table>
### get guid value

**void** `SetBlob` (const Attr::BlobAttrId &attrId, const Util::Blob &blob)

set blob value

**const Util::Blob &** `GetBlob` (const Attr::BlobAttrId &attrId) const

get blob value

**void** `SetName` (const Util::Atom<Util::String> &n)

set model node name

**void** `SetType` (ModelNodeType::Code t)

set `ModelNodeType`

**void** `SetParent` (const Ptr<ModelNode> &p)

set parent node

**void** `AddChild` (const Ptr<ModelNode> &c)

add a child node

**void** `AddVisibleNodeInstance` (IndexT frameIndex, const Ptr<ModelNodeInstance> &nodeInst)

called by model node instance on NotifyVisible()

**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
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Details</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>
<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 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 Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual <code>Ptr &lt; ModelNodeInstance &gt;</code></td>
<td><code>CreateNodeInstance ()</code> const</td>
<td>create a model node instance</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>LoadResources ()</code></td>
<td>called when resources should be loaded</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>UnloadResources ()</code></td>
<td>called when resources should be unloaded</td>
</tr>
<tr>
<td>void</td>
<td><code>SetupManagedTextureVariable</code> (const <code>Attr::AttrId</code> &amp;resAttrId, const <code>Ptr &lt; CoreGraphics::ShaderVariable &gt;</code> &amp;var)</td>
<td>setup a new managed texture variable</td>
</tr>
<tr>
<td>void</td>
<td><code>UpdateManagedTextureVariables ()</code></td>
<td>update managed texture variables</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>LoadFromAttrs</code> (const <code>Attr::AttributeContainer</code> &amp;attrs)</td>
<td>called to initialize from attributes</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>SaveToAttrs</code> (Attr::AttributeContainer &amp;attrs)</td>
<td>called to save state back to attributes</td>
</tr>
<tr>
<td>void</td>
<td><code>SetBoundingBox</code> (const <code>Math::bbox</code> &amp;b)</td>
<td>set bounding box</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnAttachToModel</code> (const <code>Ptr &lt; Model &gt;</code> &amp;model)</td>
<td>called when attached to model node</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnRemoveFromModel ()</code></td>
<td>called when removed from model node</td>
</tr>
<tr>
<td>const <code>Util::Array &lt; Ptr &lt; ModelNodeInstance &gt; &gt;</code> &amp;</td>
<td><code>GetVisibleModelNodeInstances</code> (ModelNodeType::Code t) const</td>
<td>get visible model node instances</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::StateNode::SetupManagedTextureVariable ( const Attr::AttrId & resAttrId,
                                                     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 ( ) [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
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

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><code>ShapeNodeInstance ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~ShapeNodeInstance ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual void Render ()</code></td>
<td>Perform rendering</td>
</tr>
<tr>
<td><code>virtual void ApplyState ()</code></td>
<td>Apply per-instance state prior to rendering</td>
</tr>
<tr>
<td><code>Ptr &lt; CoreGraphics::ShaderVariableInstance &gt;</code></td>
<td>Instantiate a shader variable by semantic</td>
</tr>
<tr>
<td><code>bool HasShaderVariableInstance (const CoreGraphics::ShaderVariable::Semantic &amp;semantic)</code></td>
<td>Return true if a shader variable has been instantiated</td>
</tr>
<tr>
<td><code>const Ptr &lt; CoreGraphics::ShaderVariableInstance &gt; &amp; GetShaderVariableInstance (const CoreGraphics::ShaderVariable::Semantic &amp;semantic)</code></td>
<td>Get a shader variable instance</td>
</tr>
<tr>
<td><code>virtual void Update ()</code></td>
<td>Perform per-frame updates</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 SetRotate (const Math::quaternion &amp;)</code></td>
<td>Set rotate quaternion</td>
</tr>
<tr>
<td><code>const Math::quaternion &amp; GetRotate () 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>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</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 () const</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 () const</code></td>
<td>get scale pivot</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetLocalTransform ()</code></td>
<td>get resulting local transform matrix in local parent</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp; GetModelTransform () const</code></td>
<td>get model space transform (valid after <code>Update()</code>)</td>
</tr>
<tr>
<td><code>bool HasParent () const</code></td>
<td>return true if node has a parent</td>
</tr>
<tr>
<td><code>const Ptr &lt; ModelNodeInstance &gt; &amp; GetParent () const</code></td>
<td>get parent node</td>
</tr>
<tr>
<td><code>const Util::Array &lt; Ptr &lt; ModelNodeInstance &gt; &gt; &amp; GetChildren () const</code></td>
<td>get child nodes</td>
</tr>
<tr>
<td><code>bool IsAttachedToModelInstance () const</code></td>
<td>return true if attached to <code>ModelInstance</code></td>
</tr>
<tr>
<td><code>const Ptr &lt; ModelInstance &gt; &amp; GetModelInstance () const</code></td>
<td>get the <code>ModelInstance</code> we are attached to</td>
</tr>
<tr>
<td><code>const Ptr &lt; ModelNode &gt; &amp; GetModelNode () const</code></td>
<td>get the <code>ModelNode</code> we're associated with</td>
</tr>
<tr>
<td><code>virtual void SetVisible (bool b)</code></td>
<td>set visible, used by charactersystem</td>
</tr>
<tr>
<td><code>bool IsVisible () const</code></td>
<td>is visible</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

static void DumpRefCountingLeaks ()

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 void</td>
<td><code>OnNotifyVisible</code> (IndexT frameIndex)</td>
<td>notify that we are visible</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnAttachToModelInstance</code> (const <code>Ptr&lt; ModelInstance &gt;</code> &amp;inst, const <code>Ptr&lt; ModelNode &gt;</code> &amp;node, const <code>Ptr&lt; ModelNodeInstance &gt;</code> &amp;parentNodeInst)</td>
<td>called when attached to <code>ModelInstance</code></td>
</tr>
<tr>
<td>virtual void</td>
<td><code>OnRemoveFromModelInstance</code> ()</td>
<td>called when removed from <code>ModelInstance</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetParent</code> (const <code>Ptr&lt; ModelNodeInstance &gt;</code> &amp;p)</td>
<td>set parent node</td>
</tr>
<tr>
<td>void</td>
<td><code>AddChild</code> (const <code>Ptr&lt; ModelNodeInstance &gt;</code> &amp;c)</td>
<td>add a child node</td>
</tr>
<tr>
<td>virtual void</td>
<td><code>RenderDebug</code> ()</td>
<td>render node specific debug shape</td>
</tr>
<tr>
<td>void</td>
<td><code>SetChildrenVisibility</code> (ModelNodeInstance *parent, bool b)</td>
<td>set visible flag of children</td>
</tr>
</tbody>
</table>
### Member Function Documentation

```cpp
void Models::TransformNodeInstance::Update() [virtual, inherited]
```

perform per-frame updates

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.

Reimplemented from `Models::ModelNodeInstance`.

Reimplemented in `Models::CharacterNodeInstance`, `Models::ParticleSystemNodeInstance`, and `Models::SkinShapeNodeInstance`.

```cpp
void Models::ModelNodeInstance::SetVisible(bool b) [inline, virtual, inherited]
```

set visible, used by character system

Set visibility of node and its children, should not be call per frame!

Reimplemented in `Models::SkinShapeNodeInstance`.

```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

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::SkinShapeNode
Models::SkinShapeNode Class Reference

#include <skinshapenode.h>

Inheritance diagram for Models::SkinShapeNode:
Detailed Description

A ModelNode which describes a skinned shape. Since the SkinShapeNode is derived from the ShapeNode (which in turn is derived from the TransformNode), a SkinShapeNode contains all the necessary information for rendering a skinned mesh.

It supports the use of variations and skinlists for multi exchangeable animated skins and textures (all the nebula2 character3 functionalities).

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SkinShapeNode ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~SkinShapeNode ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual Ptr &lt; ModelNodeInstance &gt; CreateNodeInstance () const</td>
<td>create a model node instance</td>
</tr>
<tr>
<td>virtual void ApplySharedState ()</td>
<td>apply state shared by all my SkinShapeNodeInstances</td>
</tr>
<tr>
<td>void BeginFragments (int num)</td>
<td>begin defining mesh fragments</td>
</tr>
<tr>
<td>void SetFragGroupIndex (int fragIndex, int primGroupIndex)</td>
<td>set mesh group index of a skin fragment</td>
</tr>
<tr>
<td>int GetFragGroupIndex (int fragIndex) const</td>
<td>get mesh group index of a skin fragment</td>
</tr>
<tr>
<td>void BeginJointPalette (int fragIndex, int numJoints)</td>
<td>begin defining the joint palette of a fragment</td>
</tr>
<tr>
<td>void SetJointIndices (int fragIndex, int paletteIndex, int ji0, int ji1, int ji2, int ji3, int ji4, int ji5, int ji6, int ji7)</td>
<td>add up to 8 joint indices to the fragments joint palette</td>
</tr>
<tr>
<td>void SetJointIndex (int fragIndex, int paletteIndex, int jointIndex)</td>
<td>add a single joint index to the fragments joint palette</td>
</tr>
<tr>
<td>void EndJointPalette (int fragIndex)</td>
<td>finish adding joints to the joint palette</td>
</tr>
<tr>
<td>void EndFragments ()</td>
<td>finish defining fragments</td>
</tr>
<tr>
<td>int GetNumFragments () const</td>
<td></td>
</tr>
</tbody>
</table>
**const Util::FixedArray < Char::CharFragment > &**

**GetFragmentArray () const**

get fragment array

**int**

**GetJointPaletteSize (int fragIndex) const**

get number of joints in a fragment's joint palette

**int**

**GetJointIndex (int fragIndex, int paletteIndex) const**

get a joint index from a fragment's joint palette

**void**

**RequestLoadResources ()**

request load resources, if not loaded yet

**void**

**RequestUnloadResources ()**

request to unload our resources

**virtual Resources::Resource::State**

**GetResourceState () const**

get overall state of contained resources (Initial, Loaded, Pending, Failed, Cancelled)

**const Ptr < Resources::ManagedMesh > &**

**GetManagedMesh () const**

get managed mesh

**const Ptr < CoreGraphics::ShaderInstance > &**

**GetShaderInstance () const**

get pointer to contained shader instance

**void**

**setPosition (const Math::point &p)**

set position

**const Math::point &**

**GetPosition () const**

get position

**void**

**setRotate (const Math::quaternion &r)**

set rotate quaternion

**const Math::quaternion &**

**GetRotate () const**

get rotate quaternion

**void**

**setScale (const Math::vector &s)**

set scale

**const Math::vector &**

**getScale () const**

get scale

**void**

**setRotatePivot (const Math::point**

set rotate pivot
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const <code>Math::point</code> &amp; <code>Math::point</code></td>
<td><code>GetRotatePivot()</code> const get rotate pivot</td>
</tr>
<tr>
<td>void <code>SetScalePivot</code> (const <code>Math::point</code> &amp;p)</td>
<td>set scale pivot</td>
</tr>
<tr>
<td>const <code>Math::point</code> &amp; <code>Math::point</code></td>
<td><code>GetScalePivot()</code> const get scale pivot</td>
</tr>
<tr>
<td>const <code>Util::Atom &lt; Util::String&gt;</code> &amp;</td>
<td><code>GetName()</code> const get model node name</td>
</tr>
<tr>
<td><code>ModelNodeType::Code</code></td>
<td><code>GetType()</code> const get the <code>ModelNodeType</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>HasParent()</code> const return true if node has a parent</td>
</tr>
<tr>
<td>const <code>Ptr&lt; ModelNode &gt;</code> &amp;</td>
<td><code>GetParent()</code> const get parent node</td>
</tr>
<tr>
<td>const <code>Util::Array &lt; Ptr&lt; ModelNode &gt; &gt;</code> &amp;</td>
<td><code>GetChildren()</code> const get child nodes</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsAttachedToModel()</code> const return true if currently attached to a <code>Model</code></td>
</tr>
<tr>
<td>const <code>Ptr&lt; Model &gt;</code> &amp;</td>
<td><code>GetModel()</code> const get model this node is attached to</td>
</tr>
<tr>
<td>const <code>Math::bbox</code> &amp;</td>
<td><code>GetBoundingBox()</code> const get bounding box of model node</td>
</tr>
<tr>
<td>const <code>Attr::AttributeContainer</code> &amp;</td>
<td><code>GetAttrs()</code> const read access to model node attributes</td>
</tr>
<tr>
<td>bool</td>
<td><code>HasAttr</code> (const <code>Attr::AttrId</code> &amp;attrId) const check if model node attribute exists</td>
</tr>
<tr>
<td>void</td>
<td><code>SetAttr</code> (const <code>Attr::Attribute</code> &amp;attr) const set generic model node attribute</td>
</tr>
<tr>
<td>const <code>Attr::Attribute</code> &amp;</td>
<td><code>GetAttr</code> (const <code>Attr::AttrId</code> &amp;attrId) const get generic model node attribute</td>
</tr>
<tr>
<td>Type</td>
<td>Function Name</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>void</td>
<td>SetBool</td>
</tr>
<tr>
<td>bool</td>
<td>GetBool</td>
</tr>
<tr>
<td>void</td>
<td>SetFloat</td>
</tr>
<tr>
<td>float</td>
<td>GetFloat</td>
</tr>
<tr>
<td>void</td>
<td>SetInt</td>
</tr>
<tr>
<td>int</td>
<td>GetInt</td>
</tr>
<tr>
<td>void</td>
<td>SetString</td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td>GetString</td>
</tr>
<tr>
<td>void</td>
<td>SetFloat4</td>
</tr>
<tr>
<td>Math::float4</td>
<td>GetFloat4</td>
</tr>
<tr>
<td>void</td>
<td>SetMatrix44</td>
</tr>
<tr>
<td>const Math::matrix44 &amp;</td>
<td>GetMatrix44</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>void SetGuid</td>
<td>Set guid value</td>
</tr>
<tr>
<td>const Util::Guid &amp;</td>
<td>GetGuid (const Attr::GuidAttrId &amp;attrId) const</td>
</tr>
<tr>
<td>void SetBlob</td>
<td>Set blob value</td>
</tr>
<tr>
<td>const Util::Blob &amp;</td>
<td>GetBlob (const Attr::BlobAttrId &amp;attrId) const</td>
</tr>
<tr>
<td>void SetName</td>
<td>Set model node name</td>
</tr>
<tr>
<td>void SetType</td>
<td>Set ModelNodeType</td>
</tr>
<tr>
<td>void SetParent</td>
<td>Set parent node</td>
</tr>
<tr>
<td>void AddChild</td>
<td>Add a child node</td>
</tr>
<tr>
<td>void AddVisibleNodeInstance</td>
<td>called by model node instance on NotifyVisible()</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</td>
<td>Return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf</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) 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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>void <strong>SetupManagedTextureVariable</strong> (const Attr::AttrId &amp;resAttrId, const Ptr&lt; CoreGraphics::ShaderVariable &gt; &amp;var)</td>
<td>setup a new managed texture variable</td>
</tr>
<tr>
<td>void <strong>UpdateManagedTextureVariables</strong> ()</td>
<td>update managed texture variables</td>
</tr>
<tr>
<td>virtual void <strong>LoadFromAttrs</strong> (const Attr::AttributeContainer &amp;attrs)</td>
<td>called to initialize from attributes</td>
</tr>
<tr>
<td>virtual void <strong>SaveToAttrs</strong> (Attr::AttributeContainer &amp;attrs)</td>
<td>called to save state back to attributes</td>
</tr>
<tr>
<td>void <strong>SetBoundingBox</strong> (const Math::bbox &amp;b)</td>
<td>set bounding box</td>
</tr>
<tr>
<td>virtual void <strong>OnAttachToModel</strong> (const Ptr&lt; Model &gt; &amp;model)</td>
<td>called when attached to model node</td>
</tr>
<tr>
<td>virtual void <strong>OnRemoveFromModel</strong> ()</td>
<td>called when removed from model node</td>
</tr>
<tr>
<td>const Util::Array&lt; Ptr&lt; ModelNodeInstance &gt; &gt; &amp; <strong>GetVisibleModelNodeInstances</strong> (ModelNodeType::Code t) const</td>
<td>get visible model node instances</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::SkinShapeNode::ApplySharedState() [virtual]
```

apply state shared by all my SkinShapeNodeInstances

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 from `Models::ShapeNode`.

```cpp
void Models::SkinShapeNode::BeginFragments(int num)
```

begin defining mesh fragments

Begin defining mesh fragment. A skin mesh may be divided into several fragments to account for gfx hardware which an only render a skinned mesh with a limited number of influence objects (joints).

```cpp
void Models::SkinShapeNode::SetFragGroupIndex(int fragIndex, int primGroupIndex)
```

set mesh group index of a skin fragment

Set the mesh group index for a skin fragment.

```cpp
int Models::SkinShapeNode::GetFragGroupIndex(int fragIndex) const
```

get mesh group index of a skin fragment

Get the mesh group index for a skin fragment.
void Models::SkinShapeNode::BeginJointPalette(int fragIndex, int numJoints)

begin defining the joint palette of a fragment

Begin defining the joint palette of a skin fragment.

void Models::SkinShapeNode::SetJointIndices(int fragIndex, int paletteIndex, int j0, int j1, int j2, int j3, int j4, int j5, int j6, int j7)

add up to 8 joint indices to the fragments joint palette

Add up to 8 joints to a fragments joint palette starting at a given palette index.

void Models::SkinShapeNode::EndJointPalette(int fragIndex)

finish adding joints to the joint palette

Finish defining the joint palette of a skin fragment.

void Models::SkinShapeNode::EndFragments()

finish defining fragments

Finish defining fragments.

int Models::SkinShapeNode::GetNumFragments() const [inline]
get number of fragments

Get number of fragments.

```cpp
const Util::FixedArray<
    Char::CharFragment >& Models::SkinShapeNode::GetFragmentArray
```

get fragment array

Get fragments.

```cpp
int Models::SkinShapeNode::GetJointPaletteSize(int fragIndex ) const [inline]
```

get number of joints in a fragment's joint palette

Get joint palette size of a skin fragment.

```cpp
int Models::SkinShapeNode::GetJointIndex(int fragIndex, int paletteIndex ) const [inline]
```

get a joint index from a fragment's joint palette

Get a joint index from a fragment's joint index.

```cpp
void Models::SkinShapeNode::RequestLoadResources()
```

request load resources, if not loaded yet

Is call by instances which are getting visible.

```cpp
void Models::SkinShapeNode::RequestUnloadResources()
```

request to unload our resources

Request an unload of our resources, if no instance node references out resource, we try to unload it
void Models::SkinShapeNode::LoadResources() [protected, virtual]

called when resources should be loaded

Do not load resources by default.

Reimplemented from Models::ShapeNode.

void Models::StateNode::SetupManagedTextureVariable(const Attr::AttrId & resAttrId,
 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.

void Models::StateNode::UpdateManagedTextureVariables() [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.

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.
Models::SkinShapeNodeInstance
Models::SkinShapeNodeInstance Class Reference

#include <skinshapenodeinstance.h>

Inheritance diagram for Models::SkinShapeNodeInstance:
Detailed Description

The **SkinShapeNodeInstance** actually renders a skinned shape, and holds all the necessary per-instance state to do so.

It knows the selected visible skins an active textures for actual rendering.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SkinShapeNodeInstance()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>~SkinShapeNodeInstance()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual Update()</code></td>
<td>Check for valid resources</td>
</tr>
<tr>
<td><code>virtual Render()</code></td>
<td>Perform rendering</td>
</tr>
<tr>
<td><code>virtual SetVisible (bool b)</code></td>
<td>Set visible, used by charactersystem</td>
</tr>
<tr>
<td><code>virtual ApplyState()</code></td>
<td>Apply per-instance state prior to rendering</td>
</tr>
<tr>
<td><code>Ptr&lt; CoreGraphics::ShaderVariableInstance &gt;</code></td>
<td>Instantiate a shader variable by semantic</td>
</tr>
<tr>
<td><code>HasShaderVariableInstance (const CoreGraphics::ShaderVariable::Semantic &amp;semantic)</code></td>
<td>Return true if a shader variable has been instantiated</td>
</tr>
<tr>
<td><code>GetShaderVariableInstance (const CoreGraphics::ShaderVariable::Semantic &amp;semantic)</code></td>
<td>Get a shader variable instance</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 SetRotate (const Math::quaternion &amp;)</code></td>
<td>Set rotate quaternion</td>
</tr>
<tr>
<td><code>const Math::quaternion &amp; GetRotate () const</code></td>
<td>Get rotate quaternion</td>
</tr>
<tr>
<td><code>void SetScale (const Math::vector &amp;s)</code></td>
<td></td>
</tr>
</tbody>
</table>
**Math::vector**

- **GetScale()**
  
  `const Math::vector & GetScale() const`  
  
  get scale

- **SetRotatePivot(const Math::point &)**
  
  `void SetRotatePivot(const Math::point &)`  
  
  set rotate pivot

- **GetRotatePivot()**
  
  `const Math::point & GetRotatePivot() const`  
  
  get rotate pivot

- **SetScalePivot(const Math::point &)**
  
  `void SetScalePivot(const Math::point &)`  
  
  set scale pivot

- **GetScalePivot()**
  
  `const Math::point & GetScalePivot() const`  
  
  get scale pivot

- **GetLocalTransform()**
  
  `const Math::matrix44 & GetLocalTransform() const`  
  
  get resulting local transform matrix in local parent space

- **GetModelTransform()**
  
  `const Math::matrix44 & GetModelTransform() const`  
  
  get model space transform (valid after Update())

- **HasParent()**
  
  `bool HasParent() const`  
  
  return true if node has a parent

- **GetParent()**
  
  `const Ptr<ModelNodeInstance> & GetParent() const`  
  
  get parent node

- **GetChildren()**
  
  `const Util::Array<Ptr<ModelNodeInstance>> & GetChildren() const`  
  
  get child nodes

- **IsAttachedToModelInstance()**
  
  `bool IsAttachedToModelInstance() const`  
  
  return true if attached to ModelInstance

- **GetModelInstance()**
  
  `const Ptr<ModelInstance> & GetModelInstance() const`  
  
  get the ModelInstance we are attached to

- **GetModelNode()**
  
  `const Ptr<ModelNode> & GetModelNode() const`  
  
  get the ModelNode we're associated with

- **IsVisible()**
  
  `bool IsVisible() const`  
  
  is visible

- **GetRefCount()**
  
  `int GetRefCount() const`  
  
  get the current refcount

- **AddRef()**
  
  `void AddRef()`  
  
  increment refcount by one

- **Release()**
  
  `void Release()`  
  
  decrement refcount and destroy object if refcount is zero
<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>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>OnNotifyVisible(IndexT frameIndex)</code></td>
<td>notify that we are visible</td>
</tr>
<tr>
<td><code>RenderSkinning()</code></td>
<td>render skinned mesh</td>
</tr>
<tr>
<td><code>RenderFragment(int primGroupIndex, Char::CharJointPalette &amp;jointPalette)</code></td>
<td>render mesh fragment</td>
</tr>
<tr>
<td><code>OnAttachToModelInstance(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>OnRemoveFromModelInstance()</code></td>
<td>called when removed from <code>ModelInstance</code></td>
</tr>
<tr>
<td><code>ValidateCharacter()</code></td>
<td>validate character</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>RenderDebug()</code></td>
<td>render node specific debug shape</td>
</tr>
<tr>
<td><code>SetChildrenVisibility(ModelNodeInstance *parent, bool)</code></td>
<td>set visible flag of children</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
void Models::SkinShapeNodeInstance::SetVisible(bool b) [inline, virtual]
```

set visible, used by charactersystem

Set visibility of node and its children, should not be call per frame!

Reimplemented from `Models::ModelNodeInstance`.

```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::StateNode
Models::StateNode Class Reference

#include <statenode.h>

Inheritance diagram for Models::StateNode:

```
Core::RefCounted

Models::ModelNode

Models::TransformNode

Models::StateNode

Models::ShapeNode

Models::ParticleSystemNode  Models::SkinShapeNode
```
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><strong>virtual ~StateNode ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>virtual Resources::Resource::State GetResourceState () const</strong></td>
<td>Get overall state of contained resources (Initial, Loaded, Pending, Failed, Cancelled)</td>
</tr>
<tr>
<td><strong>virtual void ApplySharedState ()</strong></td>
<td>Apply state shared by all my ModelNodeInstances</td>
</tr>
<tr>
<td><strong>const Ptr &lt; CoreGraphics::ShaderInstance &gt; &amp; GetShaderInstance () const</strong></td>
<td>Get pointer to contained shader instance</td>
</tr>
<tr>
<td><strong>void SetPosition (const Math::point &amp;p)</strong></td>
<td>Set position</td>
</tr>
<tr>
<td><strong>const Math::point &amp; GetPosition () const</strong></td>
<td>Get position</td>
</tr>
<tr>
<td><strong>void SetRotate (const Math::quaternion &amp;r)</strong></td>
<td>Set rotate quaternion</td>
</tr>
<tr>
<td><strong>const Math::quaternion &amp; GetRotate () const</strong></td>
<td>Get rotate quaternion</td>
</tr>
<tr>
<td><strong>void SetScale (const Math::vector &amp;s)</strong></td>
<td>Set scale</td>
</tr>
<tr>
<td><strong>const Math::vector &amp; GetScale () const</strong></td>
<td>Get scale</td>
</tr>
<tr>
<td><strong>void SetRotatePivot (const Math::point &amp;p)</strong></td>
<td>Set rotate pivot</td>
</tr>
<tr>
<td><strong>const Math::point &amp; GetRotatePivot () const</strong></td>
<td>Get rotate pivot</td>
</tr>
<tr>
<td><strong>void SetScalePivot (const Math::point &amp;p)</strong></td>
<td>Set scale pivot</td>
</tr>
<tr>
<td><strong>const Math::point &amp;</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>GetScalePivot () const</code></td>
<td>Get scale pivot</td>
</tr>
<tr>
<td><code>const Util::Atom &lt; Util::String &gt; &amp; GetName () const</code></td>
<td>Get model node name</td>
</tr>
<tr>
<td><code>const ModelNodeType::Code GetType () const</code></td>
<td>Get the <code>ModelNodeType</code></td>
</tr>
<tr>
<td><code>bool HasParent () const</code></td>
<td>Return true if node has a parent</td>
</tr>
<tr>
<td><code>const Ptr&lt; ModelNode &gt; &amp; GetParent () const</code></td>
<td>Get parent 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 IsAttachedToModel () const</code></td>
<td>Return true if currently attached to a <code>Model</code></td>
</tr>
<tr>
<td><code>const Ptr&lt; Model &gt; &amp; GetModel () const</code></td>
<td>Get model this node is attached to</td>
</tr>
<tr>
<td><code>const Math::bbox &amp; GetBoundingBox () const</code></td>
<td>Get bounding box of model node</td>
</tr>
<tr>
<td><code>const Attr::AttributeContainer &amp; GetAttrs () const</code></td>
<td>Read access to model node attributes</td>
</tr>
<tr>
<td><code>bool HasAttr (const Attr::AttrId &amp;attrId) const</code></td>
<td>Check if model node attribute exists</td>
</tr>
<tr>
<td><code>void SetAttr (const Attr::Attribute &amp;attr)</code></td>
<td>Set generic model node attribute</td>
</tr>
<tr>
<td><code>const Attr::Attribute &amp; GetAttr (const Attr::AttrId &amp;attrId) const</code></td>
<td>Get generic model node attribute</td>
</tr>
<tr>
<td><code>void SetBool (const Attr::BoolAttrId &amp;attrId, bool val)</code></td>
<td>Set bool value</td>
</tr>
<tr>
<td><code>bool GetBool (const Attr::BoolAttrId &amp;attrId) const</code></td>
<td>Get bool value</td>
</tr>
<tr>
<td><code>void SetFloat (const Attr::FloatAttrId</code></td>
<td>Set float value</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>float</td>
<td><code>GetFloat (const Attr::FloatAttrId &amp;attrId)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetInt (const Attr::IntAttrId &amp;attrId, int val)</code></td>
</tr>
<tr>
<td>int</td>
<td><code>GetInt (const Attr::IntAttrId &amp;attrId)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetString (const Attr::StringAttrId &amp;attrId, const Util::String &amp;val)</code></td>
</tr>
<tr>
<td>const Util::String &amp;</td>
<td><code>GetString (const Attr::StringAttrId &amp;attrId)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetFloat4 (const Attr::Float4AttrId &amp;attrId, const Math::float4 &amp;val)</code></td>
</tr>
<tr>
<td>Math::float4</td>
<td><code>GetFloat4 (const Attr::Float4AttrId &amp;attrId)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetMatrix44 (const Attr::Matrix44AttrId &amp;attrId, const Math::matrix44 &amp;val)</code></td>
</tr>
<tr>
<td>const Math::matrix44 &amp;</td>
<td><code>GetMatrix44 (const Attr::Matrix44AttrId &amp;attrId)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetGuid (const Attr::GuidAttrId &amp;attrId, const Util::Guid &amp;guid)</code></td>
</tr>
<tr>
<td>const Util::Guid &amp;</td>
<td><code>GetGuid (const Attr::GuidAttrId &amp;attrId)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>SetBlob (const Attr::BlobAttrId &amp;attrId)</code></td>
</tr>
</tbody>
</table>
&attrId, const Util::Blob &blob)
set blob value

const Util::Blob &
GetBlob (const Attr::BlobAttrId &attrId) const
get blob value

void SetName (const Util::Atom<Util::String> &n)
set model node name

void SetType (ModelNodeType::Code t)
set ModelNodeType

void SetParent (const Ptr<ModelNode> &p)
set parent node

void AddChild (const Ptr<ModelNode> &c)
add a child node

void AddVisibleNodeInstance (IndexT frameIndex, const Ptr<ModelNodeInstance> &nodeInst)
called by model node instance on NotifyVisible()

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
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>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>virtual <code>Ptr &lt; ModelNodeInstance &gt;</code> <code>CreateNodeInstance ()</code> const</td>
<td><code>create a model node instance</code></td>
</tr>
<tr>
<td>virtual <code>void</code> <code>LoadResources ()</code></td>
<td><code>called when resources should be loaded</code></td>
</tr>
<tr>
<td>virtual <code>void</code> <code>UnloadResources ()</code></td>
<td><code>called when resources should be unloaded</code></td>
</tr>
<tr>
<td><code>void</code> <code>SetupManagedTextureVariable (const Attr::AttrId &amp;resAttrId, const </code></td>
<td></td>
</tr>
<tr>
<td><code>Ptr &lt; CoreGraphics::ShaderVariable &gt; &amp;var)</code></td>
<td><code>setup a new managed texture variable</code></td>
</tr>
<tr>
<td><code>void</code> <code>UpdateManagedTextureVariables ()</code></td>
<td><code>update managed texture variables</code></td>
</tr>
<tr>
<td>virtual <code>void</code> <code>LoadFromAttrs (const Attr::AttributeContainer &amp;attrs)</code></td>
<td><code>called to initialize from attributes</code></td>
</tr>
<tr>
<td><code>virtual</code> <code>void</code> <code>SaveToAttrs (Attr::AttributeContainer &amp;attrs)</code></td>
<td><code>called to save state back to attributes</code></td>
</tr>
<tr>
<td><code>void</code> <code>SetBoundingBox (const Math::bbox &amp;b)</code></td>
<td><code>set bounding box</code></td>
</tr>
<tr>
<td>virtual <code>void</code> <code>OnAttachToModel (const </code></td>
<td></td>
</tr>
<tr>
<td><code>Ptr &lt; Model &gt; &amp;model)</code></td>
<td><code>called when attached to model node</code></td>
</tr>
<tr>
<td><code>virtual</code> <code>void</code> <code>OnRemoveFromModel ()</code></td>
<td><code>called when removed from model node</code></td>
</tr>
<tr>
<td><code>const Util::Array &lt; </code></td>
<td></td>
</tr>
<tr>
<td><code>Ptr &lt; ModelNodeInstance &gt; &gt; &amp;</code> <code>GetVisibleModelNodeInstances (ModelNodeType::Code t)</code></td>
<td><code>get visible model node instances</code></td>
</tr>
</tbody>
</table>
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.

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.
Models::StateNodeInstance
Models::StateNodeInstance Class Reference

#include <statenodeinstance.h>

Inheritance diagram for Models::StateNodeInstance:
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;</code> CreateShaderVariableInstance (const CoreGraphics::ShaderVariable::Semantic &amp;semantic)`</td>
<td>instanciate a shader variable by semantic</td>
</tr>
<tr>
<td>bool HasShaderVariableInstance (const CoreGraphics::ShaderVariable::Semantic &amp;semantic) const</td>
<td>return true if a shader variable has been instanteated</td>
</tr>
<tr>
<td>const <code>Ptr &lt; CoreGraphics::ShaderVariableInstance &gt;</code> &amp; GetShaderVariableInstance (const CoreGraphics::ShaderVariable::Semantic &amp;semantic) const</td>
<td>get a shader variable instance</td>
</tr>
<tr>
<td>virtual void <code>Update ()</code></td>
<td>perform per-frame updates</td>
</tr>
<tr>
<td>void <code>setPosition</code> (const Math::point &amp;p)`</td>
<td>set position</td>
</tr>
<tr>
<td>const Math::point &amp; <code>GetPosition ()</code> const</td>
<td>get position</td>
</tr>
<tr>
<td>void <code>setRotate</code> (const Math::quaternion &amp;q)`</td>
<td>set rotate quaternion</td>
</tr>
<tr>
<td>const Math::quaternion &amp; <code>getRotate ()</code> const</td>
<td>get rotate quaternion</td>
</tr>
<tr>
<td>void <code>setScale</code> (const Math::vector &amp;s)`</td>
<td>set scale</td>
</tr>
<tr>
<td>const Math::vector &amp; <code>getScale ()</code> const</td>
<td>get scale</td>
</tr>
<tr>
<td>void <code>setRotatePivot</code> (const Math::point &amp;p)`</td>
<td>set rotate pivot</td>
</tr>
</tbody>
</table>
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

const Math::matrix44 & GetLocalTransform ()
get resulting local transform matrix in local parent

const Math::matrix44 & GetModelTransform () const
get model space transform (valid after Update())

virtual void Render ()
perform rendering

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 IsAttachedToModelInstance () const
return true if attached to ModelInstance

const Ptr< ModelInstance > & GetModelInstance () const
get the ModelInstance we are attached to

const Ptr< ModelNode > & GetModelNode () const
get the ModelNode we’re associated with

virtual void SetVisible (bool b)
set visible, used by charactersystem

bool IsVisible () const
is visible

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one

void Release ()
decrement refcount and destroy object if refcount is zero
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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></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>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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>virtual void OnAttachToModelInstance (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>virtual void OnRemoveFromModelInstance ()</code></td>
<td>Called when removed from <code>ModelInstance</code></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>
<tr>
<td><code>virtual void OnNotifyVisible (IndexT frameIndex)</code></td>
<td>Notify that we are visible</td>
</tr>
<tr>
<td><code>virtual void RenderDebug ()</code></td>
<td>Render node specific debug shape</td>
</tr>
<tr>
<td><code>void SetChildrenVisibility (ModelNodeInstance *parent, bool)</code></td>
<td>Set visible flag of children</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::TransformNodeInstance::Update() [virtual, inherited]
```

perform per-frame updates

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.

Reimplemented from `Models::ModelNodeInstance`.

Reimplemented in `Models::CharacterNodeInstance`, `Models::ParticleSystemNodeInstance`, and `Models::SkinShapeNodeInstance`.

```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 `Models::ParticleSystemNodeInstance`, `Models::ShapeNodeInstance`, and `Models::SkinShapeNodeInstance`.

```cpp
void Models::ModelNodeInstance::SetVisible(bool b) [inline, virtual, inherited]
```

set visible, used by charactersystem

Set visibility of node and its children, should not be call per frame!

Reimplemented in `Models::SkinShapeNodeInstance`. 
void Models::ModelNodeInstance::OnNotifyVisible(IndexT frameIndex) [protected, virtual, inherited]

notify that we are visible

This method is called from the NotifyVisible() method of our ModelInstance object. If the ModelNodeInstance provides something renderable it should respond by adding itself as visible node instance to its model node.

Reimplemented in Models::ParticleSystemNodeInstance, Models::ShapeNodeInstance, and Models::SkinShapeNodeInstance.

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.

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

dump 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

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 OnLoadRequested() 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>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</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>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>
</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
int Core::RefCounted::GetRefCount(const [inline, inherited]

get the current refcount

Return the current refcount of the object.

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

increment refcount by one

Increment the refcount of the object.

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

dump 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><strong>TransformNode</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~TransformNode</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>setPosition</strong></td>
<td>Set position</td>
</tr>
<tr>
<td><strong>GetPosition</strong></td>
<td>Get position</td>
</tr>
<tr>
<td><strong>set rotate</strong></td>
<td>Set rotate quaternion</td>
</tr>
<tr>
<td><strong>GetRotate</strong></td>
<td>Get rotate quaternion</td>
</tr>
<tr>
<td><strong>setScale</strong></td>
<td>Set scale</td>
</tr>
<tr>
<td><strong>GetScale</strong></td>
<td>Get scale</td>
</tr>
<tr>
<td><strong>set rotate pivot</strong></td>
<td>Set rotate pivot</td>
</tr>
<tr>
<td><strong>GetRotatePivot</strong></td>
<td>Get rotate pivot</td>
</tr>
<tr>
<td><strong>setScale pivot</strong></td>
<td>Set scale pivot</td>
</tr>
<tr>
<td><strong>GetScalePivot</strong></td>
<td>Get scale pivot</td>
</tr>
<tr>
<td>virtual <strong>ApplySharedState</strong></td>
<td>Apply state shared by all my ModelNodeInstances</td>
</tr>
<tr>
<td>virtual <strong>GetResourceState</strong></td>
<td>Get overall state of contained resources (Initial, Loaded, Pending, Failed, Cancelled)</td>
</tr>
<tr>
<td><strong>getName</strong></td>
<td>Get model node name</td>
</tr>
<tr>
<td><strong>getType</strong></td>
<td>Get model node type code</td>
</tr>
</tbody>
</table>
get the `ModelNodeType`

bool `HasParent` () const
return true if node has a parent

const `Ptr< ModelNode >` & `GetParent` () const
get parent node

const `Util::Array< Ptr< ModelNode > >` & `GetChildren` () const
get child nodes

bool `IsAttachedToModel` () const
return true if currently attached to a `Model`

const `Ptr< Model >` & `GetModel` () const
get model this node is attached to

const `Math::bbox` & `GetBoundingBox` () const
get bounding box of model node

const `Attr::AttributeContainer` & `GetAttrs` () const
read access to model node attributes

bool `HasAttr` (const `Attr::AttrId` &attrId) const
check if model node attribute exists

void `SetAttr` (const `Attr::Attribute` &attr)
set generic model node attribute

const `Attr::Attribute` & `GetAttr` (const `Attr::AttrId` &attrId) const
get generic model node attribute

void `SetBool` (const `Attr::BoolAttrId` &attrId, bool val)
set bool value

bool `GetBool` (const `Attr::BoolAttrId` &attrId) const
get bool value

void `SetFloat` (const `Attr::FloatAttrId` &attrId, float val)
set float value

float `GetFloat` (const `Attr::FloatAttrId` &attrId) const
get float value

void `SetInt` (const `Attr::IntAttrId` &attrId, int val)
set int value

int `GetInt` (const `Attr::IntAttrId` &attrId) const
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>get int value</code></td>
<td></td>
</tr>
<tr>
<td><code>void SetString (const Attr::StringAttrId &amp;attrId, const Util::String &amp;val)</code></td>
<td>set string value</td>
</tr>
<tr>
<td><code>const Util::String &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>const Util::String &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>GetString (const Attr::StringAttrId &amp;attrId)</code></td>
<td>get string value</td>
</tr>
<tr>
<td><code>void SetFloat4 (const Attr::Float4AttrId &amp;attrId, const Math::float4 &amp;val)</code></td>
<td>set float4 value</td>
</tr>
<tr>
<td><code>Math::float4</code></td>
<td></td>
</tr>
<tr>
<td><code>Math::float4</code></td>
<td></td>
</tr>
<tr>
<td><code>GetFloat4 (const Attr::Float4AttrId &amp;attrId)</code></td>
<td>get float4 value</td>
</tr>
<tr>
<td><code>void SetMatrix44 (const Attr::Matrix44AttrId &amp;attrId, const Math::matrix44 &amp;val)</code></td>
<td>set matrix44 value</td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>const Math::matrix44 &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>GetMatrix44 (const Attr::Matrix44AttrId &amp;attrId)</code></td>
<td>get matrix44 value</td>
</tr>
<tr>
<td><code>void SetGuid (const Attr::GuidAttrId &amp;attrId, const Util::Guid &amp;guid)</code></td>
<td>set guid value</td>
</tr>
<tr>
<td><code>const Util::Guid &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>const Util::Guid &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>GetGuid (const Attr::GuidAttrId &amp;attrId)</code></td>
<td>get guid value</td>
</tr>
<tr>
<td><code>void SetBlob (const Attr::BlobAttrId &amp;attrId, const Util::Blob &amp;blob)</code></td>
<td>set blob value</td>
</tr>
<tr>
<td><code>const Util::Blob &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>const Util::Blob &amp;</code></td>
<td></td>
</tr>
<tr>
<td><code>GetBlob (const Attr::BlobAttrId &amp;attrId)</code></td>
<td>get blob value</td>
</tr>
<tr>
<td><code>void SetName (const Util::Atom&lt; Util::String &gt; &amp;n)</code></td>
<td>set model node name</td>
</tr>
<tr>
<td><code>void SetType (ModelNodeType::Code t)</code></td>
<td>set <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>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>set parent node</td>
<td>add a child node</td>
</tr>
<tr>
<td><code>AddChild</code> (const <code>Ptr&lt;ModelNode&gt;</code> &amp;c)</td>
<td>called by model node instance on <code>NotifyVisible()</code></td>
</tr>
<tr>
<td><code>AddVisibleNodeInstance</code> (IndexT frameIndex, const <code>Ptr&lt;ModelNodeInstance&gt;</code> &amp;nodeInst)</td>
<td></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>
<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><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</strong> <code>Ptr &lt; ModelNodeInstance &gt;</code> <strong>CreateNodeInstance</strong> () const</td>
<td>create a model node instance</td>
</tr>
<tr>
<td><code>LoadFromAttrs</code> (const <code>Attr::AttributeContainer</code> &amp;attrs)</td>
<td>called to initialize from attributes</td>
</tr>
<tr>
<td><code>SaveToAttrs</code> (<code>Attr::AttributeContainer</code> &amp;attrs)</td>
<td>called to save state back to attributes</td>
</tr>
<tr>
<td><code>SetBoundingBox</code> (const <code>Math::bbox</code> &amp;b)</td>
<td>set bounding box</td>
</tr>
<tr>
<td><strong>virtual void</strong> <code>OnAttachToModel</code> (const <code>Ptr &lt; Model &gt;</code> &amp;model)</td>
<td>called when attached to model node</td>
</tr>
<tr>
<td><strong>virtual void</strong> <code>OnRemoveFromModel</code> ()</td>
<td>called when removed from model node</td>
</tr>
<tr>
<td><strong>virtual void</strong> <code>LoadResources</code> ()</td>
<td>called when resources should be loaded</td>
</tr>
<tr>
<td><strong>virtual void</strong> <code>UnloadResources</code> ()</td>
<td>called when resources should be unloaded</td>
</tr>
<tr>
<td><code>const Util::Array &lt; Ptr &lt; ModelNodeInstance &gt; &gt; &amp; GetVisibleModelNodeInstances (ModelNodeType::Code t)</code> const</td>
<td>get visible model node instances</td>
</tr>
</tbody>
</table>
### Member Function Documentation

#### void Models::ModelNode::ApplySharedState() [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 `Models::ShapeNode`, `Models::SkinShapeNode`, and `Models::StateNode`.

#### Resource::State Models::ModelNode::GetResourceState() const [virtual, inherited]

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 `Models::CharacterNode`, `Models::ShapeNode`, and `Models::StateNode`.

#### void Models::ModelNode::LoadResources() [protected, virtual, inherited]

called when resources should be loaded

This method is called when required resources should be loaded.
Reimplemented in `Models::CharacterNode`, `Models::ParticleSystemNode`, `Models::ShapeNode`, `Models::SkinShapeNode`, and `Models::StateNode`.

```cpp
void Models::ModelNode::UnloadResources() [protected, virtual, inherited]
```
called when resources should be unloaded

This method is called when required resources should be unloaded.

Reimplemented in `Models::CharacterNode`, `Models::ParticleSystemNode`, `Models::ShapeNode`, and `Models::StateNode`.

```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 Tue Feb 19 12:16:51 2008
Models::TransformNodeInstance
Models::TransformNodeInstance Class Reference

#include <transformnodeinstance.h>

Inheritance diagram for Models::TransformNodeInstance:

Core::RefCounted

Models::ModelNodeInstance

Models::TransformNodeInstance

Models::CharacterNodeInstance    Models::StateNodeInstance

Models::ShapeNodeInstance

Models::ParticleSystemNodeInstance    Models::SkinShapeNodeInstance
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>virtual ~TransformNodeInstance()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>virtual void Update()</code></td>
<td>Perform per-frame updates</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><code>void SetRotatePivot(const Math::point &amp;p)</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;p)</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>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 Render()</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>bool HasParent() const</td>
<td>return true if node has a parent</td>
</tr>
<tr>
<td>const Ptr&lt;ModelNodeInstance&gt; &amp; GetParent() const</td>
<td>get parent node</td>
</tr>
<tr>
<td>const Util::Array&lt;Ptr&lt;ModelNodeInstance&gt;&gt; &amp; GetChildren() const</td>
<td>get child nodes</td>
</tr>
<tr>
<td>bool IsAttachedToModelInstance() const</td>
<td>return true if attached to ModelInstance</td>
</tr>
<tr>
<td>const Ptr&lt;ModelInstance&gt; &amp; GetModelInstance() const</td>
<td>get the ModelInstance we are attached to</td>
</tr>
<tr>
<td>const Ptr&lt;ModelNode&gt; &amp; GetModelNode() const</td>
<td>get the ModelNode we’re associated with</td>
</tr>
<tr>
<td>virtual void SetVisible(bool b)</td>
<td>set visible, used by charactersystem</td>
</tr>
<tr>
<td>bool IsVisible() const</td>
<td>is visible</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></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 string</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>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>virtual void OnAttachToModelInstance(const $\text{Ptr&lt; ModelInstance &gt;}$ &amp;inst, const $\text{Ptr&lt; ModelNode &gt;}$ &amp;node, const $\text{Ptr&lt; ModelNodeInstance &gt;}$ &amp;parentNodeInst)</strong></td>
<td>called when attached to <code>ModelInstance</code></td>
</tr>
<tr>
<td><strong>virtual void OnRemoveFromModelInstance()</strong></td>
<td>called when removed from <code>ModelInstance</code></td>
</tr>
<tr>
<td><strong>void SetParent(const $\text{Ptr&lt; ModelNodeInstance &gt;}$ &amp;p)</strong></td>
<td>set parent node</td>
</tr>
<tr>
<td><strong>void AddChild(const $\text{Ptr&lt; ModelNodeInstance &gt;}$ &amp;c)</strong></td>
<td>add a child node</td>
</tr>
<tr>
<td><strong>virtual void OnNotifyVisible(IndexT frameIndex)</strong></td>
<td>notify that we are visible</td>
</tr>
<tr>
<td><strong>virtual void RenderDebug()</strong></td>
<td>render node specific debug shape</td>
</tr>
<tr>
<td>*<em>void SetChildrenVisibility(ModelNodeInstance <em>parent, bool b)</em></em></td>
<td>set visible flag of children</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::TransformNodeInstance::Update() [virtual]
```

perform per-frame updates

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.

Reimplemented from `Models::ModelNodeInstance`.

Reimplemented in `Models::CharacterNodeInstance`, `Models::ParticleSystemNodeInstance`, and `Models::SkinShapeNodeInstance`.

```cpp
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`.

```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()` invokation.

Reimplemented in `Models::ParticleSystemNodeInstance`,
Models::ShapeNodeInstance, and
Models::SkinShapeNodeInstance.

void
Models::ModelNodeInstance::SetVisible (bool b) [inline, virtual, inherited]

set visible, used by charactersystem

Set visibility of node and its children, should not be call per frame!

Reimplemented in Models::SkinShapeNodeInstance.

void
Models::ModelNodeInstance::OnNotifyVisible (IndexT frameIndex) [protected, virtual, inherited]

notify that we are visible

This method is called from the NotifyVisible() method of our ModelInstance object. If the ModelNodeInstance provides something renderable it should respond by adding itself as visible node instance to its model node.

Reimplemented in Models::ParticleSystemNodeInstance, Models::ShapeNodeInstance, and Models::SkinShapeNodeInstance.

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::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>VisResolveContainer ()</td>
<td>constructor</td>
</tr>
<tr>
<td>void Reset ()</td>
<td>reset content</td>
</tr>
<tr>
<td>void SetResolved (ModelNodeType::Code t, bool b)</td>
<td>set the resolved flag for a given ModelNodeType</td>
</tr>
<tr>
<td>bool IsResolved (ModelNodeType::Code t) const</td>
<td>return true if the resolved flag has been set</td>
</tr>
<tr>
<td>void Add (IndexT frameIndex, ModelNodeType::Code t, const Ptr&lt; TYPE &gt; &amp;e)</td>
<td>add a visible element by ModelNodeType</td>
</tr>
<tr>
<td>const Util::Array&lt; Ptr&lt; TYPE &gt; &gt; &amp; Get (ModelNodeType::Code t) 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:

```
Core::RefCounted

Models::VisResolver
```

Detailed Description

The VisResolver accepts visible ModellInstances 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>VisResolver ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~VisResolver ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void Open ()</strong></td>
<td>open the visibility resolver</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close the visibility resolver</td>
</tr>
<tr>
<td><strong>bool IsOpen () const</strong></td>
<td>return true if currently open</td>
</tr>
<tr>
<td><strong>void BeginResolve ()</strong></td>
<td>begin resolving</td>
</tr>
<tr>
<td><strong>void AttachVisibleModelInstance (const Ptr&lt;ModelInstance&gt; &amp;inst)</strong></td>
<td>attach a visible ModelInstance</td>
</tr>
<tr>
<td><strong>void EndResolve ()</strong></td>
<td>end resolve</td>
</tr>
<tr>
<td><strong>const Util::Array&lt; Ptr&lt;Model&gt; &gt; &amp; GetVisibleModels (ModelNodeType::Code nodeType) const</strong></td>
<td>post-resolve: get Models with visible ModelNodeInstances by node type</td>
</tr>
<tr>
<td><strong>const Util::Array&lt; Ptr&lt;ModelNode&gt; &gt; &amp; GetVisibleModelNodes (ModelNodeType::Code nodeType, const Ptr&lt;Model&gt; &amp;model) const</strong></td>
<td>post-resolve: get visible ModelNodes of a Model by node type</td>
</tr>
<tr>
<td><strong>const Util::Array&lt; Ptr&lt;ModelNodeInstance&gt; &gt; &amp; GetVisibleModelNodeInstances (ModelNodeType::Code nodeType, const Ptr&lt;ModelNode&gt; &amp;modelNode) const</strong></td>
<td>post-resolve: get visible ModelNodeInstance of a ModelNode by node type</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></td>
</tr>
</tbody>
</table>
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

\begin{verbatim}
static void DumpRefCountingLeaks ()
   dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)
\end{verbatim}
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.
Models::XmlModelReader
Models::XmlModelReader Class Reference

#include <xmlmodelreader.h>

Inheritance diagram for Models::XmlModelReader:
Detailed Description

Implements the ModelReader for the human-readable XML file format.

(C) 2007 Radon Labs GmbH
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XmlModelReader ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <strong>~XmlModelReader ()</strong></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>virtual bool FillModel ()</td>
<td>parse resource and build model hierarchy</td>
</tr>
<tr>
<td>void SetModelResId (const Resources::ResourceId &amp;resId)</td>
<td>set the read cursor to the first Model in the stream</td>
</tr>
<tr>
<td>const Resources::ResourceId &amp; GetModelResId () const</td>
<td>get the model resource id</td>
</tr>
<tr>
<td>void SetModel (const Ptr&lt; Model &gt; &amp;model)</td>
<td>get attributes of current model</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>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>
</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, 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

const 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 (NEBUŁA3_DEBUG builds only!)</em></td>
<td></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Models::ModelReader::SetModelResId(const Resources::ResourceId &resId) [inherited]
```

set the read cursor to the first `Model` in the stream

set the read cursor to the next `Model` in the stream set model resource id (some file format's don't provide their own)

```cpp
void Models::ModelReader::SetModel(const Ptr<Model> &model) [inline, inherited]
```

get attributes of current model

set the read cursor to the first `ModelNode` in the current `Model` set the read cursor to the next `ModelNode` in the current `Model` get the model node name get model node class get the name of the parent node get attributes of current model node set model for filling up with modelnodes and attributes

```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 \texttt{HasStream()} to determine if a stream is attached.

\begin{verbatim}
bool
IO::StreamReader::HasStream( ) const [inherited]

return true if a stream is set
\end{verbatim}

Returns true if a stream is attached to the reader.

\begin{verbatim}
int
Core::RefCounted::GetRefCount( ) const [inline, inherited]

get the current refcount
\end{verbatim}

Return the current refcount of the object.

\begin{verbatim}
void
Core::RefCounted::AddRef( ) [inline, inherited]

increment refcount by one
\end{verbatim}

Increment the refcount of the object.

\begin{verbatim}
void
Core::RefCounted::Release( ) [inline, inherited]

decrement refcount and destroy object if refcount is zero
\end{verbatim}

Decrement the refcount and destroy object if refcount is zero.

\begin{verbatim}
const Util::String &
Core::RefCounted::GetClassName( ) const [inline, inherited]

get the class name
\end{verbatim}

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.

```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::XmlModelWriter
Models::XmlModelWriter Class Reference

#include <xmlmodelwriter.h>

Inheritance diagram for Models::XmlModelWriter:
Detailed Description

Implements the **ModelWriter** class for the human-readable XML file format.

(C) 2007 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>XmlModelWriter ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <code>~XmlModelWriter ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>virtual <code>Util::String GetFileExtension () const</code></td>
<td>get the file extension used by the writer</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>virtual bool <code>BeginModel (const Ptr&lt; Model &gt; &amp;model)</code></td>
<td>begin writing a new Model</td>
</tr>
<tr>
<td>virtual void <code>WriteModelAttrs (const Attr::AttributeContainer &amp;attrs)</code></td>
<td>write model attributes</td>
</tr>
<tr>
<td>virtual void <code>EndModel ()</code></td>
<td>end writing current Model</td>
</tr>
<tr>
<td>virtual bool <code>BeginModelNode (const Ptr&lt; ModelNode &gt; &amp;modelNode)</code></td>
<td>begin writing a new ModelNode</td>
</tr>
<tr>
<td>virtual void <code>WriteModelNodeAttrs (const Attr::AttributeContainer &amp;attrs)</code></td>
<td>write mode node attributes</td>
</tr>
<tr>
<td>virtual void <code>EndModelNode ()</code></td>
<td>end writing current ModelNode</td>
</tr>
<tr>
<td>void <code>SetStream (const Ptr&lt; Stream &gt; &amp;s)</code></td>
<td>set stream to write to</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Stream &gt; &amp; GetStream () const</code></td>
<td>get currently set stream</td>
</tr>
<tr>
<td>bool <code>HasStream () const</code></td>
<td>return true if a stream is set</td>
</tr>
<tr>
<td>bool <code>IsOpen () const</code></td>
<td>return true if currently open</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

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.

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

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::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>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><code>Open (Protocol p)</code></td>
<td>open the socket</td>
</tr>
<tr>
<td>void</td>
<td><code>Close ()</code></td>
<td>close the socket</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsOpen () const</code></td>
<td>return true if the socket is open</td>
</tr>
<tr>
<td>ErrorCode</td>
<td><code>GetErrorCode () const</code></td>
<td>get the last error code</td>
</tr>
<tr>
<td>Util::String</td>
<td><code>GetErrorString () const</code></td>
<td>get the last error string</td>
</tr>
<tr>
<td>void</td>
<td><code>SetAddress (const Net::IpAddress &amp;a)</code></td>
<td>set internet address of socket</td>
</tr>
<tr>
<td>const Net::IpAddress &amp;</td>
<td><code>GetAddress () const</code></td>
<td>get internet address of socket</td>
</tr>
<tr>
<td>void</td>
<td><code>SetBroadcast (bool b)</code></td>
<td>set the broadcast flag (SO_BROADCAST)</td>
</tr>
<tr>
<td>bool</td>
<td><code>GetBroadcast ()</code></td>
<td>get the broadcast flag</td>
</tr>
<tr>
<td>void</td>
<td><code>SetDontLinger (bool b)</code></td>
<td>set the don't linger flag (SO_DONTLINGER)</td>
</tr>
<tr>
<td>bool</td>
<td><code>GetDontLinger ()</code></td>
<td>get the don't linger flag</td>
</tr>
<tr>
<td>void</td>
<td><code>SetKeepAlive (bool b)</code></td>
<td>set the keepalive flag (SO_KEEPALIVE)</td>
</tr>
<tr>
<td>bool</td>
<td><code>GetKeepAlive ()</code></td>
<td>get the keepalive flag</td>
</tr>
<tr>
<td>void</td>
<td><code>SetReUseAddr (bool b)</code></td>
<td>set reuseaddr flag (SO_REUSEADDR)</td>
</tr>
<tr>
<td>bool</td>
<td><code>GetReUseAddr ()</code></td>
<td>get reuseaddr flag</td>
</tr>
<tr>
<td>void</td>
<td><code>SetNoDelay (bool b)</code></td>
<td>set nodelay flag (TCP_NODELAY)</td>
</tr>
<tr>
<td>bool</td>
<td><code>GetNoDelay ()</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><code>SetRecvBufSize</code> (SizeT s)</td>
<td>void, set receive buffer size</td>
<td></td>
</tr>
<tr>
<td><code>GetRecvBufSize</code> ()</td>
<td>SizeT, get receive buffer size</td>
<td></td>
</tr>
<tr>
<td><code>SetSendBufSize</code> (SizeT s)</td>
<td>void, set send buffer size</td>
<td></td>
</tr>
<tr>
<td><code>GetSendBufSize</code> ()</td>
<td>SizeT, get send buffer size</td>
<td></td>
</tr>
<tr>
<td><code>SetBlocking</code> (bool b)</td>
<td>void, set blocking mode (FIONBIO)</td>
<td></td>
</tr>
<tr>
<td><code>GetBlocking</code> () const</td>
<td>bool, get blocking mode</td>
<td></td>
</tr>
<tr>
<td><code>GetMaxMsgSize</code> ()</td>
<td>SizeT, get the maximum message size that can be sent atomically</td>
<td></td>
</tr>
<tr>
<td><code>Bind</code> ()</td>
<td>bool, bind socket to ip address</td>
<td></td>
</tr>
<tr>
<td><code>IsBound</code> () const</td>
<td>bool, return true if the socket is bound to an address</td>
<td></td>
</tr>
<tr>
<td><code>Listen</code> ()</td>
<td>bool, listen for incoming connections (for server sockets)</td>
<td></td>
</tr>
<tr>
<td><code>Accept</code> (Ptr<a href="">Net::Socket</a> &amp;outSocket)</td>
<td>bool, accept incoming connection, return a new socket (for server sockets)</td>
<td></td>
</tr>
<tr>
<td><code>Connect</code> ()</td>
<td>Result, connect to the sockets address (for client sockets)</td>
<td></td>
</tr>
<tr>
<td><code>IsConnected</code> ()</td>
<td>bool, test if the socket is currently connected</td>
<td></td>
</tr>
<tr>
<td><code>Send</code> (const void *buf, SizeT numBytes, SizeT &amp;bytesSent)</td>
<td>Result, send raw data into the socket</td>
<td></td>
</tr>
<tr>
<td><code>HasRecvData</code> ()</td>
<td>bool, return true if recv data is available at the socket</td>
<td></td>
</tr>
<tr>
<td><code>Recv</code> (void *buf, SizeT bufSize, SizeT &amp;bytesReceived)</td>
<td>Result, receive raw data from the socket</td>
<td></td>
</tr>
<tr>
<td><code>SendTo</code> (const void *buf, SizeT numBytes, uint...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**send raw data to address for connectionless sockets**

**recv raw data from address for connectionless sockets**

get the current refcount

increment refcount by one

decrement refcount and destroy object if refcount is zero

return true if this object is instance of given class

return true if this object is instance of given class by string

return true if this object is instance of given class by fourcc

return true if this object is instance of given class, or a derived class

return true if this object is instance of given class, or a derived class, by string

return true if this object is instance of given class, or a derived class, by fourcc

get the class name

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

void Win32::Win32Socket::SetAddress (Net::IpAddress a) [inline, inherited]

set internet address of socket
Set internet address of socket.

const Net::IpAddress & Win32::Win32Socket::GetAddress () const [inline, inherited]

get internet address of socket
Get internet address of socket.

void Win32::Win32Socket::SetBlocking (bool b) [inherited]

set blocking mode (FIONBIO)
Set the socket to blocking mode.

bool Win32::Win32Socket::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.

bool Win32::Win32Socket::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.
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".

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 established.

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.

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.

```cpp
bool Win32::Win32Socket::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.

```cpp
Win32Socket::Result Win32::Win32Socket::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.

```cpp
Win32Socket::Result Win32::Win32Socket::SendTo( const void * buf,
```
send raw data to address for connectionless sockets

FIXME: this is the send method for connectionless sockets using the UDP protocol.

Win32Socket::Result Win32::Win32Socket::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.

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:

```
Core::RefCounted

Net::TcpClient
```

---
Detailed Description

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.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TcpClient ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~TcpClient ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>SetBlocking (bool b)</strong></td>
<td>enable/disable blocking behaviour</td>
</tr>
<tr>
<td><strong>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 () const</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>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</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>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

TcpClient::Result
Net::TcpClient::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.

void
Net::TcpClient::Disconnect()

disconnect from the server

This disconnects the current connection.

bool
Net::TcpClient::IsConnected() const

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.

```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 Tue Feb 19 12:16:52 2008
Net::TcpClientConnection
Net::TcpClientConnection Class Reference

#include <tcpclientconnection.h>

Inheritance diagram for Net::TcpClientConnection:
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 theRecvStream. 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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TcpClientConnection ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~TcpClientConnection ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>bool <strong>Connect</strong> (const Ptr&lt; Socket &gt; &amp;s)</td>
<td>connect using provided socket</td>
</tr>
<tr>
<td>bool <strong>IsConnected</strong> () const</td>
<td>get the connection status</td>
</tr>
<tr>
<td>void <strong>Shutdown</strong> ()</td>
<td>shutdown the connection</td>
</tr>
<tr>
<td>const IpAddress &amp; <strong>GetClientAddress</strong> () const</td>
<td>get the client’s ip address</td>
</tr>
<tr>
<td>Socket::Result <strong>Send</strong> ()</td>
<td>send accumulated content of send stream to server</td>
</tr>
<tr>
<td>const Ptr&lt; IO::Stream &gt; &amp; <strong>GetSendStream</strong> ()</td>
<td>access to send stream</td>
</tr>
<tr>
<td>Socket::Result <strong>Recv</strong> ()</td>
<td>receive data from server into recv stream</td>
</tr>
<tr>
<td>const Ptr&lt; IO::Stream &gt; &amp; <strong>GetRecvStream</strong> ()</td>
<td>access to recv stream</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 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>
<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
This method should be called as the very last before an application exits.
Net::TcpServer
Net::TcpServer Class Reference

#include <tcpserver.h>

Inheritance diagram for Net::TcpServer:

```
Core::RefCounted

Net::TcpServer
```


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>TcpServer ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>virtual ~TcpServer ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>void SetAddress (const IpAddress &amp;addr)</strong></td>
<td>set address, hostname can be “any”, “self” or “inetself”</td>
</tr>
<tr>
<td><strong>const IpAddress &amp; getAddress () const</strong></td>
<td>get address</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 () const</strong></td>
<td>return true if server is 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) 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>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</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>
</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.
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
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

PhysicsFeature::MouseGripperProperty
PhysicsFeature::MouseGrippingProperty
Class Reference

#include <mousegrippingproperty.h>

---
Detailed Description

(C) 2005 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 ~MouseGripperProperty ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual void SetupCallbacks ()</td>
<td>setup callbacks for this property</td>
</tr>
<tr>
<td>virtual void SetupDefaultAttributes ()</td>
<td>setup default entity attributes</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 OnBeginFrame ()</td>
<td>called after movement has happened</td>
</tr>
<tr>
<td>virtual void OnMoveBefore ()</td>
<td>called before movement has happened</td>
</tr>
<tr>
<td>virtual void OnMoveAfter ()</td>
<td>called after movement has happened</td>
</tr>
<tr>
<td>virtual void OnRenderDebug ()</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**
PhysicsFeature::PhysicsProperty Class Reference

#include <physicsproperty.h>

Inherited by PhysicsFeature::ActorPhysicsProperty.
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) 2005 Radon Labs GmbH
Public Member Functions

- **PhysicsProperty ()**
  - constructor

- **virtual ~PhysicsProperty ()**
  - destructor

- **virtual void SetupCallbacks ()**
  - setup callbacks for this property

- **virtual void SetupDefaultAttributes ()**
  - setup default entity attributes

- **virtual void OnActivate ()**
  - called from Entity::ActivateProperties()

- **virtual void OnDeactivate ()**
  - called from Entity::DeactivateProperties()

- **virtual void SetupAcceptedMessages ()**
  - override to register accepted messages

- **virtual void handleMessage (const Ptr<Messaging::Message> &msg)**
  - handle a single message

- **virtual void OnMoveAfter ()**
  - called after movement has happened

- **virtual Physics::PhysicsEntity * GetPhysicsEntity () const**
  - get a pointer to the physics entity

- **void SetEnabled (bool enabled)**
  - enable/disable physics

- **bool IsEnabled () const**
  - is physics enabled
## 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>EnablePhysics ()</td>
<td>enable and activate the physics, overload in subclass</td>
</tr>
<tr>
<td>virtual void</td>
<td>DisablePhysics ()</td>
<td>disable and cleanup the physics, overload in subclass</td>
</tr>
<tr>
<td>void</td>
<td>ApplyImpulseAtPos (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>
</tbody>
</table>
Member Function Documentation

```cpp
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.

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.

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.

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!

void PhysicsFeature::PhysicsProperty::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
Apply an impulse vector at a position in the global coordinate frame.
#include <pidfeedbackloop.h>
Detailed Description

A PID feedback loop (proportional integral derivative feedback loop)

(C) 2007 RadonLabs GmbH
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

#include <pquatfeedbackloop.h>
Detailed Description

A specialized proportional feedback loop for rotations, using a quaternion representation.

(C) 2007 RadonLabs GmbH
PreShaders::BoxFilterKernel
PreShaders::BoxFilterKernel Class Reference

#include <boxfilterkernel.h>


Detailed Description

Implements a simple pre-shader which writes a 2x2 box filter kernel for generating mipmaps.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnAttach</strong></td>
<td>(const Ptr&lt; CoreGraphics::ShaderInstance &gt;&amp;shdInst)</td>
</tr>
<tr>
<td></td>
<td>called when attached to shader instance</td>
</tr>
<tr>
<td><strong>OnDetach</strong></td>
<td>()</td>
</tr>
<tr>
<td></td>
<td>called when detached from shader instance</td>
</tr>
<tr>
<td><strong>OnApply</strong></td>
<td>()</td>
</tr>
<tr>
<td></td>
<td>called before rendering the shader</td>
</tr>
</tbody>
</table>
PreShaders::GaussianBlur5x5FilterKernel
PreShaders::GaussianBlur5x5FilterKernel
Class Reference

#include <gaussianblur5x5filterkernel.h>
Detailed Description

Implement a pre-shader which writes a 5x5 Gaussian blur filter kernel to the shader.

(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 OnAttach (const Ptr&lt; CoreGraphics::ShaderInstance &gt;&amp;shdInst)</code></td>
<td>called when attached to shader instance</td>
</tr>
<tr>
<td><code>virtual void OnDetach ()</code></td>
<td>called when detached from shader instance</td>
</tr>
<tr>
<td><code>virtual void OnApply ()</code></td>
<td>called before rendering the shader</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:52 2008
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
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>Setup</code></td>
<td>Setup the object</td>
</tr>
<tr>
<td><code>Reset</code></td>
<td>Reset the object to its default settings</td>
</tr>
<tr>
<td><code>Update</code></td>
<td>Update the view matrix</td>
</tr>
<tr>
<td><code>GetCameraTransform()</code></td>
<td>Get the current camera transform</td>
</tr>
<tr>
<td><code>SetOrbitButton(bool b)</code></td>
<td>Set state of orbit button</td>
</tr>
<tr>
<td><code>SetPanButton(bool b)</code></td>
<td>Set state of panning button</td>
</tr>
<tr>
<td><code>SetZoomButton(bool b)</code></td>
<td>Set state of zoom button</td>
</tr>
<tr>
<td><code>SetZoomInButton(bool b)</code></td>
<td>Set state of zoom-in button</td>
</tr>
<tr>
<td><code>SetZoomOutButton(bool b)</code></td>
<td>Set state of zoom-out button</td>
</tr>
<tr>
<td><code>SetMouseMovement(const Math::float2 &amp;v)</code></td>
<td>Set mouse movement</td>
</tr>
<tr>
<td><code>SetZoomIn(float v)</code></td>
<td>Set zoom-in value</td>
</tr>
<tr>
<td><code>SetZoomOut(float v)</code></td>
<td>Set zoom-out value</td>
</tr>
<tr>
<td><code>SetPanning(const Math::float2 &amp;v)</code></td>
<td>Set panning vector</td>
</tr>
<tr>
<td><code>SetOrbiting(const Math::float2 &amp;v)</code></td>
<td>Set orbiting vector</td>
</tr>
</tbody>
</table>
Resources::DynamicMeshResourceLoader
Resources::DynamicMeshResourceLoader
Class Reference

#include <dynamicmeshresourceloader.h>

Inheritance diagram for Resources::DynamicMeshResourceLoader:
Detailed Description

A resource loader for dynamic meshes.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynamicMeshResourceLoader ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~DynamicMeshResourceLoader ()</td>
<td>Destructor</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>virtual bool OnLoadRequested ()</td>
<td>called by resource when a load is requested</td>
</tr>
<tr>
<td>void SetNumberOfIndices (SizeT num)</td>
<td>sets number of indices, if 0 no index buffer allocated</td>
</tr>
<tr>
<td>void SetNumberOfVertices (SizeT num)</td>
<td>sets number of vertices</td>
</tr>
<tr>
<td>SizeT GetNumberOfIndices () const</td>
<td>gets the number of vertices</td>
</tr>
<tr>
<td>SizeT GetNumberOfVertices () const</td>
<td>gets the number of indices</td>
</tr>
<tr>
<td>void SetIndexData (void *data)</td>
<td>sets index data</td>
</tr>
<tr>
<td>void SetVertexData (void *data)</td>
<td>sets vertex data</td>
</tr>
<tr>
<td>void SetIndexUsage (CoreGraphics::IndexBuffer::Usage usage)</td>
<td>sets the usage of the index buffer</td>
</tr>
<tr>
<td>void SetVertexUsage (CoreGraphics::VertexBuffer::Usage usage)</td>
<td>sets the usage of the vertex buffer</td>
</tr>
<tr>
<td>void SetIndexAccessMode (CoreGraphics::IndexBuffer::Access accessMode)</td>
<td>sets the access mode of the index buffer</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void SetVertexAccessMode</code></td>
<td>Sets the access mode of the vertex buffer</td>
</tr>
<tr>
<td><code>void SetIndexBufferType</code></td>
<td>Sets the index buffer type</td>
</tr>
<tr>
<td><code>void SetVertexComponents (const</code></td>
<td>Sets the vertex buffer components</td>
</tr>
<tr>
<td><code>Util::Array&lt;CoreGraphics::VertexComponent&gt;</code> &amp;components)</td>
<td></td>
</tr>
<tr>
<td><code>void SetVertexWidth</code> (int vertexWidth)</td>
<td>Sets the width of a vertex</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>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>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><strong>IsA</strong> (const <strong>Rtti</strong>&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><strong>IsA</strong> (const <strong>Util::String</strong>&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><strong>IsA</strong> (const <strong>Util::FourCC</strong>&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 <strong>Util::String</strong>&amp;</th>
<th><strong>GetClassName</strong> () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>get the class name</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Util::FourCC</strong></th>
<th><strong>GetClassFourCC</strong> () 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>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::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, Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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 Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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 `Direct3D9::D3D9StreamTextureLoader`, `CoreGraphics::StreamAnimationLoader`, `CoreGraphics::StreamMeshLoader`, and `Models::StreamModelLoader`.

```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.
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
### Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>Priority</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>priority levels</em></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 (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 <code>UpdateRenderStats()</code>)</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 current priority</td>
</tr>
<tr>
<td><code>Priority GetPriority () const</code></td>
<td>get the current priority</td>
</tr>
<tr>
<td><code>Resource::State GetState () const</code></td>
<td>get current resource loading state (Initial, Pending, Loaded, Failed, Cancelled)</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Resource &gt;</code> &amp; GetResource () const</td>
<td>get contained resource (may return placeholder)</td>
</tr>
<tr>
<td>void Clear ()</td>
<td>clear the contained resource</td>
</tr>
<tr>
<td>void SetResource (const <code>Ptr&lt; Resource &gt;</code> &amp;resource)</td>
<td>set actual resource</td>
</tr>
<tr>
<td>void SetPlaceholder (const <code>Ptr&lt; Resource &gt;</code> &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 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

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(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.

```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::ManagedResource**
Resources::ManagedResource Class Reference

#include <managedresource.h>

Inheritance diagram for Resources::ManagedResource:

```
Core::RefCounted

Resources::ManagedResource

Models::ManagedModel  Resources::ManagedMesh  Resources::ManagedTexture
```
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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>priority levels</td>
</tr>
</tbody>
</table>
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManagedResource ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~ManagedResource ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void ClearRenderStats ()</td>
<td>clear render statistics</td>
</tr>
<tr>
<td>void UpdateRenderStats (const Math::float2 &amp;screenSpaceSize)</td>
<td>update render statistics (called by client)</td>
</tr>
<tr>
<td>void SetResourceId (const ResourceId &amp;id)</td>
<td>set resource id</td>
</tr>
<tr>
<td>const ResourceId &amp; GetResourceId () const</td>
<td>get resource id</td>
</tr>
<tr>
<td>void SetResourceType (const Core::Rtti *rtti)</td>
<td>set contained resource type</td>
</tr>
<tr>
<td>const Core::Rtti * GetResourceType () const</td>
<td>get contained resource type</td>
</tr>
<tr>
<td>void IncrClientCount ()</td>
<td>increment client count</td>
</tr>
<tr>
<td>void DecrClientCount ()</td>
<td>decrement client count</td>
</tr>
<tr>
<td>SizeT GetClientCount () const</td>
<td>get current client count</td>
</tr>
<tr>
<td>SizeT GetRenderCount () const</td>
<td>get render count for this frame (number of calls to UpdateRenderStats())</td>
</tr>
<tr>
<td>const Math::float2 &amp; GetMaxScreenSpaceSize ()</td>
<td>get maximum screen space size this frame</td>
</tr>
<tr>
<td>void 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></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetResource()</code> const</td>
<td>get contained resource (may return placeholder)</td>
</tr>
<tr>
<td><code>Clear()</code></td>
<td>clear the contained resource</td>
</tr>
<tr>
<td><code>SetResource(const Ptr&lt; Resource &gt;&amp; resource)</code></td>
<td>set actual resource</td>
</tr>
<tr>
<td><code>SetPlaceholder(const Ptr&lt; Resource &gt;&amp; placeholder)</code></td>
<td>set placeholder 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 &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 name</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><em>dump refcounting leaks, call at end of application (NEBULA3_DEBUG builds only!)</em></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(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
ResourceManager 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(

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.

\texttt{\textnormal{const } \textnormal{Util::String }\& \textnormal{Core::RefCounted::GetClassName ( ) const [inline, inherited]}}

get the class name

Get the class name of the object.

\texttt{\textnormal{Util::FourCC \ Core::RefCounted::GetClassFourCC ( ) const [inline, inherited]}}

get the class FourCC code

Get the class FourCC of the object.

\texttt{\textnormal{void \ Core::RefCounted::DumpRefCountingLeaks ( ) [static, inherited]}}

dump 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::ManagedTexture
Resources::ManagedTexture Class Reference

#include <managedtexture.h>

Inheritance diagram for Resources::ManagedTexture:
Detailed Description

A specialized managed resource for texture resources.

(C) 2007 Radon Labs GmbH
Public Types

<table>
<thead>
<tr>
<th>enum</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>priority levels</td>
</tr>
</tbody>
</table>


<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</td>
</tr>
<tr>
<td><code>void ClearRenderStats ()</code></td>
<td>clear render statistics</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 UpdateRenderStats)</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 current priority</td>
</tr>
<tr>
<td><code>Priority GetPriority () const</code></td>
<td>get the current priority</td>
</tr>
<tr>
<td><code>Resource::State GetState () const</code></td>
<td>get current resource loading state (Initial, Pending, Loaded, Failed, Cancelled)</td>
</tr>
<tr>
<td>const <code>Ptr&lt; Resource &gt;</code> &amp;</td>
<td>GetResource () const</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>void</td>
<td>Clear ()</td>
</tr>
<tr>
<td>void</td>
<td>SetResource (const <code>Ptr&lt; Resource &gt;</code> &amp;resource)</td>
</tr>
<tr>
<td>void</td>
<td>SetPlaceholder (const <code>Ptr&lt; Resource &gt;</code> &amp;placeholder)</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>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 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 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).

```cpp
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.

```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::Resource
#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 **SharedResourceServer** 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>
<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>Resource ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~Resource ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>SetAsyncEnabled (bool b)</strong></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>SetResourceId (const ResourceId &amp;id)</strong></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 (const Ptr&lt;ResourceLoader&gt; &amp;loader)</strong></td>
<td>set optional resource loader</td>
</tr>
<tr>
<td>const Ptr&lt;ResourceLoader&gt; &amp; <strong>GetLoader ()</strong> const</td>
<td>get optional resource loader</td>
</tr>
<tr>
<td>void <strong>SetSaver (const Ptr&lt;ResourceSaver&gt; &amp;saver)</strong></td>
<td>set optional resource saver</td>
</tr>
<tr>
<td>const Ptr&lt;ResourceSaver&gt; &amp; <strong>GetSaver ()</strong> const</td>
<td>get optional resource saver</td>
</tr>
<tr>
<td>SizeT <strong>GetUseCount ()</strong> const</td>
<td>get current use count</td>
</tr>
<tr>
<td>virtual <strong>State Load ()</strong></td>
<td>load the resource</td>
</tr>
<tr>
<td>virtual void <strong>Unload ()</strong></td>
<td>unload the resource, or cancel the pending load</td>
</tr>
<tr>
<td><strong>GameState ()</strong> const</td>
<td>get current state</td>
</tr>
<tr>
<td>bool <strong>IsLoaded ()</strong> const</td>
<td>return true if current state is Loaded</td>
</tr>
<tr>
<td>bool <strong>IsPending ()</strong> const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>return true if current state is Pending</code></td>
<td>bool LoadFailed () const</td>
</tr>
<tr>
<td></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>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

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>void SetState (State S)</td>
<td>set current state</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>
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 Base::MeshBase, Base::VertexBufferBase, CoreGraphics::CPUIndexBuffer, CoreGraphics::CPUVertexBuffer, Direct3D9::D3D9IndexBuffer, Direct3D9::D3D9Shader, Direct3D9::D3D9Texture, Direct3D9::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::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:

- Core::RefCounted
  - Resources::ResourceLoader
    - Base::MemoryIndexBufferLoaderBase
    - Base::MemoryVertexBufferLoaderBase
    - CoreGraphics::StreamAnimationLoader
    - CoreGraphics::StreamMeshLoader
    - Direct3D::D3D9StreamShaderLoader
    - Direct3D::D3D9StreamTextureLoader
    - Models::StreamModelLoader
    - Resources::DynamicMeshResourceLoader
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>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ResourceLoader ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>~ResourceLoader ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><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><strong>OnRemoveFromResource ()</strong></td>
<td>Called when the resource loader is removed from its resource</td>
</tr>
<tr>
<td><strong>IsAttachedToResource () const</strong></td>
<td>Return true if attached to resource</td>
</tr>
<tr>
<td><strong>GetResource () const</strong></td>
<td>Get pointer to resource</td>
</tr>
<tr>
<td><strong>CanLoadAsync () const</strong></td>
<td>Return true if asynchronous loading is supported</td>
</tr>
<tr>
<td><strong>OnLoadRequested ()</strong></td>
<td>Called by resource when a load is requested</td>
</tr>
<tr>
<td><strong>OnLoadCancelled ()</strong></td>
<td>Called by resource to cancel a pending load</td>
</tr>
<tr>
<td><strong>OnPending ()</strong></td>
<td>Call frequently while after OnLoadRequested() to put Resource into loaded state</td>
</tr>
<tr>
<td><strong>GetState () const</strong></td>
<td>Return current state</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></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const 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>
</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
```
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, Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, and Models::StreamModelLoader.

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::CPUMemoryIndexBufferLoader, CoreGraphics::CPUMemoryVertexBufferLoader, Direct3D9::D3D9MemoryIndexBufferLoader, Direct3D9::D3D9MemoryVertexBufferLoader, Direct3D9::D3D9StreamShaderLoader, Direct3D9::D3D9StreamTextureLoader,
CoreGraphics::StreamAnimationLoader,
CoreGraphics::StreamMeshLoader, Models::StreamModelLoader,
and Resources::DynamicMeshResourceLoader.

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 Direct3D9::D3D9StreamTextureLoader,
CoreGraphics::StreamAnimationLoader,
CoreGraphics::StreamMeshLoader, and
Models::StreamModelLoader.

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 Direct3D9::D3D9StreamTextureLoader,
CoreGraphics::StreamAnimationLoader,
CoreGraphics::StreamMeshLoader, and
Models::StreamModelLoader.

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]
```

ingcrement 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.
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.

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

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

<table>
<thead>
<tr>
<th>Member Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ResourceManager ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~ResourceManager ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void Open ()</td>
<td>open the resource manager</td>
</tr>
<tr>
<td>void Close ()</td>
<td>close the resource manager</td>
</tr>
<tr>
<td>bool IsOpen () const</td>
<td>return true if resource manager is open</td>
</tr>
<tr>
<td>void AttachMapper (const Ptr&lt;ResourceMapper&gt;&amp; mapper)</td>
<td>register a resource mapper (resource type is defined by mapper)</td>
</tr>
<tr>
<td>void RemoveMapper (const Core::Rtti&amp; resourceType)</td>
<td>unregister a resource mapper by resource type</td>
</tr>
<tr>
<td>void RemoveAllMappers ()</td>
<td>unregister all mappers</td>
</tr>
<tr>
<td>bool HasMapper (const Core::Rtti&amp; resourceType)</td>
<td>return true if a mapper has been registered for the given resource type</td>
</tr>
<tr>
<td>const Ptr&lt;ResourceMapper&gt;&amp; GetMapperByResourceType (const Core::Rtti&amp; resourceType)</td>
<td>get the resource mapper registered with a resource type</td>
</tr>
<tr>
<td>Ptr&lt;ManagedResource&gt; CreateManagedResource (const Core::Rtti&amp; resType, const Resourceld&amp;id)</td>
<td>create a ManagedResource object (bumps usecount on existing resource)</td>
</tr>
<tr>
<td>void DiscardManagedResource (const Ptr&lt;ManagedResource&gt; &amp;managedResource)</td>
<td>unregister a ManagedResource object</td>
</tr>
<tr>
<td>bool HasManagedResource (const Resourceld</td>
<td></td>
</tr>
</tbody>
</table>
const Ptr<ManagedResource> &

LookupManagedResource (const Resourceld &id) const

return true if a managed resource exists

return a managed resource (does not change usecount of resource)

Prepare ()

prepare stats gathering, call per frame

Update ()

perform actual resource management, call per frame

GetRefCount () const

get the current refcount

AddRef ()

increment refcount by one

Release ()

decrement refcount and destroy object if refcount is zero

IsInstanceOf (const Rtti &rtti) const

return true if this object is instance of given class

IsInstanceOf (const Util::String &className) const

return true if this object is instance of given class by string

IsInstanceOf (const Util::FourCC &classFourCC) const

return true if this object is instance of given class by fourcc

IsA (const Rtti &rtti) const

return true if this object is instance of given class, or a derived class

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 &rttiFourCC) const

return true if this object is instance of given class, or a derived class, by fourcc

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!)
```
**Member Function Documentation**

```cpp
Ptr< ManagedResource >
Resources::ResourceManager::CreateManagedResource(
    const Core::Rtti & resType,
    const ResourceId & resId
)
```

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()
```

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()
```

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
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.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Resources::ResourceMapper
Resources::ResourceMapper Class Reference

#include <resourcemapper.h>

Inheritance diagram for Resources::ResourceMapper:

```
Core::RefCounted
   |
   v
Resources::ResourceMapper
       |
       v
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><code>ResourceMapper ()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual <code>~ResourceMapper ()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td>void <code>SetPlaceholderResourceId (const ResourceId &amp;resId)</code></td>
<td>Set a placeholder resource id</td>
</tr>
<tr>
<td>const ResourceId &amp; <code>GetPlaceholderResourceId ()</code> const</td>
<td>Get placeholder resource id</td>
</tr>
<tr>
<td>void <code>SetAsyncEnabled (bool b)</code></td>
<td>Set asynchronous behaviour (default is asynchronous)</td>
</tr>
<tr>
<td>bool <code>IsAsyncEnabled ()</code> const</td>
<td>Return asynchronous loading state</td>
</tr>
<tr>
<td>virtual const Core::Rtti &amp; <code>GetResourceType ()</code> const</td>
<td>Get resource type handled by this resource mapper</td>
</tr>
<tr>
<td>virtual void <code>OnAttachToResourceManager ()</code></td>
<td>Called from resource manager when mapper is attached</td>
</tr>
<tr>
<td>virtual void <code>OnRemoveFromResourceManager ()</code></td>
<td>Called from resource manager when mapper is removed</td>
</tr>
<tr>
<td>bool <code>IsAttachedToResourceManager ()</code> const</td>
<td>Return true if currently attached to server</td>
</tr>
<tr>
<td>virtual Ptr &lt; ManagedResource &gt; <code>OnCreateManagedResource (const Core::Rtti &amp;resType, const ResourceId &amp;resId)</code></td>
<td>Called when a managed resource should be created</td>
</tr>
<tr>
<td>virtual void <code>OnDiscardManagedResource (const Ptr&lt; ManagedResource &gt; &amp;managedResource)</code></td>
<td>Called when a managed resource should be discarded</td>
</tr>
<tr>
<td>virtual void <code>OnPrepare ()</code></td>
<td>Called before gathering render stats</td>
</tr>
<tr>
<td>virtual void <code>OnUpdate ()</code></td>
<td>Called after gathering render stats to perform resource management</td>
</tr>
<tr>
<td>int <code>GetRefCount ()</code> const</td>
<td></td>
</tr>
</tbody>
</table>
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

```plaintext
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::ResourceSaver
Resources::ResourceSaver Class Reference

#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>Constructor</td>
</tr>
<tr>
<td><strong>~ResourceSaver ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>virtual void 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>virtual void OnRemoveFromResource ()</strong></td>
<td>Called when the resource saver 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 OnSave ()</strong></td>
<td>Called by resource when a save is requested</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</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>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
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::SharedResourceServer
Resources::SharedResourceServer
Class Reference

#include <sharedresourceserver.h>

Inheritance diagram for Resources::SharedResourceServer:
Detailed Description

The **SharedResourceServer** manages the pool of shared resources.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SharedResourceServer ()</strong></td>
<td>Constructor</td>
</tr>
<tr>
<td><strong>virtual ~SharedResourceServer ()</strong></td>
<td>Destructor</td>
</tr>
<tr>
<td><strong>bool Open ()</strong></td>
<td>open the resource server</td>
</tr>
<tr>
<td><strong>void Close ()</strong></td>
<td>close the resource server, discards resources</td>
</tr>
<tr>
<td><strong>bool IsOpen () const</strong></td>
<td>return true if resource server is open</td>
</tr>
<tr>
<td><strong>bool HasSharedResource (const ResourceId &amp;id) const</strong></td>
<td>return true if a shared resource exists</td>
</tr>
<tr>
<td><strong>const Ptr&lt; Resource &gt; &amp;</strong> LookupSharedResource (const ResourceId &amp;id) const**</td>
<td>lookup a shared resource</td>
</tr>
<tr>
<td><strong>Ptr&lt; Resource &gt;</strong> CreateSharedResource (const ResourceId &amp;id, const Core::Rtti &amp;resClass)**</td>
<td>create and register a shared resource</td>
</tr>
<tr>
<td><strong>Ptr&lt; Resource &gt;</strong> CreateSharedResource (const ResourceId &amp;id, const Core::Rtti &amp;resClass, const Core::Rtti &amp;loaderClass)**</td>
<td>create and register shared resource with associated loader</td>
</tr>
<tr>
<td><strong>Ptr&lt; Resource &gt;</strong> CreateSharedResource (const ResourceId &amp;id, const Core::Rtti &amp;resClass, const Core::Rtti &amp;loaderClass, const Core::Rtti &amp;saverClass)**</td>
<td>create and register shared resource with associated loader and saver</td>
</tr>
<tr>
<td><strong>void RegisterSharedResource (const Ptr&lt; Resource &gt; &amp;res)</strong></td>
<td>register an existing resource object as shared resource</td>
</tr>
<tr>
<td><strong>void UnregisterSharedResource (const Ptr&lt; Resource &gt; &amp;res)</strong></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>UnregisterSharedResource</code></td>
<td>unregister a shared resource by resource name</td>
</tr>
<tr>
<td><code>GetSharedResources</code></td>
<td>read-only access to shared resources</td>
</tr>
<tr>
<td><code>GetSharedResourcesByType</code></td>
<td>get shared resources by type (slow)</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>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>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>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 name</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

Ptr< Resource >
Resources::SharedResourceServer::CreateSharedResource
(const ResourceId resId, &
const Core::Rtti resClass &
)

create and register a shared resource

Create a shared 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 UnregisterSharedResource() when the resource is no longer needed in order to manage the use count properly.

Ptr< Resource >
Resources::SharedResourceServer::CreateSharedResource
(const ResourceId resId, &
const Core::Rtti resClass, &
const Core::Rtti loaderClass &
)

create and register shared resource with associated loader

Create and register a shared resource object with attached resource loader.
create and register shared resource with associated loader and saver

Create a shared resource object with attached resource loader and saver.

```cpp
void Resources::SharedResourceServer::RegisterSharedResource (const Ptr<Resource> & res )
```

register an existing resource object as shared resource

Register an existing resource object as shared 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::SharedResourceServer::UnregisterSharedResource (const Ptr<Resource> & res )
```

unregister a shared resource (necessary for managing the use count)

Unregister a shared 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 shared resource pool).

```cpp
void Resources::SharedResourceServer::UnregisterSharedResource (ResourceId id )
```

unregister a shared resource by resource name

Unregister a shared resource by resource id.

```cpp
Array<Ptr<Resource>> Resources::SharedResourceServer::GetSharedResourcesByVersion (Core::Rtti type ) const
```
get shared resources by type (slow)

Returns an array of shared resources by type. This is a slow 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.

```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::SimpleResourceMapper**
Resources::SimpleResourceMapper
Class Reference

#include <simpleresourcemapper.h>

Inheritance diagram for Resources::SimpleResourceMapper:

```
                Core::RefCounted
                   |
                   v
Resources::ResourceMapper
                   |
                   v
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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SimpleResourceMapper</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~SimpleResourceMapper</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td>void <strong>SetResourceClass</strong> (const Core::Rtti &amp;resClass)</td>
<td>config: set resource type handled by resource mapper (e.g. Mesh::RTTI)</td>
</tr>
<tr>
<td>void <strong>SetResourceLoaderClass</strong> (const Core::Rtti &amp;resLoaderClass)</td>
<td>config: set resource loader class (e.g. Resource::RTTI)</td>
</tr>
<tr>
<td>void <strong>SetManagedResourceClass</strong> (const Core::Rtti &amp;managedResClass)</td>
<td>config: set managed resource type handled by resource (e.g. ManagedMesh::RTTI)</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>virtual Ptr&lt;ManagedResource&gt; <strong>OnCreateManagedResource</strong> (const Core::Rtti &amp;resType, const Resourceld &amp;resId)</td>
<td>called when a managed resource should be created</td>
</tr>
<tr>
<td>virtual void <strong>OnDiscardManagedResource</strong> (const Ptr&lt;ManagedResource&gt; &amp;managedResource)</td>
<td>called when a managed resource should be discarded</td>
</tr>
<tr>
<td>virtual void <strong>OnPrepare</strong> ()</td>
<td>called before gathering render stats</td>
</tr>
<tr>
<td>virtual void <strong>OnUpdate</strong> ()</td>
<td>called after gathering render stats to perform resource management</td>
</tr>
</tbody>
</table>
const Resourceld & GetPlaceholderResourceld () const  
get placeholder resource id

void SetAsyncEnabled (bool b)  
set asynchronous behaviour (default is asynchronous)

bool IsAsyncEnabled () const  
return asynchronous loading state

bool IsAttachedToResourceManager () const  
return true if currently attached to server

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

**Ptr< ManagedResource >**

Resources::SimpleResourceMapper::OnCreateManagedResource

```cpp
const Core::Rtti resType,
const ResourceId resId
)
```  

[virtual]

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**.

**void**

Resources::SimpleResourceMapper::OnDiscardManagedResource

```cpp
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**.

**void**

Resources::SimpleResourceMapper::OnPrepare

```cpp
()
```  

[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
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 Tue Feb 19 12:16:52 2008
Scripting::Arg
Scripting::Arg Class Reference

#include <arg.h>
Detailed Description

An argument for scripting, this is just a typedef from `Util::Variant`.

(C) 2006 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Scripting::ArgsBlock
Scripting::ArgsBlock Class Reference

#include <argsblock.h>
Detailed Description

Used to pass arguments into a script command and to pass the result of the command back.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgsBlock()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~ArgsBlock()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>bool IsEmpty()</code> const</td>
<td>return true if the argument block is empty</td>
</tr>
<tr>
<td><code>void Clear()</code></td>
<td>clear the argument block</td>
</tr>
<tr>
<td><code>void AddArg()</code></td>
<td>add an argument to the block</td>
</tr>
<tr>
<td><code>SizeT GetNumArgs()</code> const</td>
<td>return number of arguments</td>
</tr>
<tr>
<td><code>bool HasArg()</code> const</td>
<td>return true if argument exists by name</td>
</tr>
<tr>
<td><code>const Util::String &amp;</code> GetArgName(IndexT index)` const</td>
<td>get argument name at index</td>
</tr>
<tr>
<td><code>const Arg &amp;</code> GetArgValue(IndexT index)` const</td>
<td>get argument value at index (read-only)</td>
</tr>
<tr>
<td><code>const Arg &amp;</code> GetArgValue(const Util::String &amp;name)` const</td>
<td>get argument value by name (read-only)</td>
</tr>
<tr>
<td><code>Arg &amp;</code> ArgValue(IndexT index)`</td>
<td>argument value at index (read/write)</td>
</tr>
<tr>
<td><code>Arg &amp;</code> ArgValue(const Util::String &amp;name)`</td>
<td>argument value by name (read/write)</td>
</tr>
</tbody>
</table>
add an argument to the block

Add argument to the args block. This method is usually called in the OnRegister() method of a script command object to setup the arguments and results block of the command.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Scripting::Command
Scripting::Command Class Reference

#include <command.h>

Inheritance diagram for Scripting::Command:

```
Core::RefCounted

Scripting::Command
```
Detailed Description

**Base** class for script commands. A script command object implements a new scripting command in a language independent fashion. New script commands are added by deriving a new class from **Scripting::Command** and registering it with the script server.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command</strong> ()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~<strong>Command</strong> ()</td>
<td>Destructor</td>
</tr>
<tr>
<td>virtual void <strong>OnRegister</strong> ()</td>
<td>called when the scripting command is registered</td>
</tr>
<tr>
<td>virtual void <strong>OnUnregister</strong> ()</td>
<td>called when the scripting command is unregistered</td>
</tr>
<tr>
<td>bool <strong>IsRegistered</strong> () const</td>
<td>return true if currently registered</td>
</tr>
<tr>
<td>virtual bool <strong>OnExecute</strong> ()</td>
<td>called when the script command is executed</td>
</tr>
<tr>
<td>virtual <strong>Util::String</strong> <strong>GetHelp</strong> () const</td>
<td>return a short help string, describing the command</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetName</strong> () const</td>
<td>get the name under which the command has been registered</td>
</tr>
<tr>
<td><strong>Util::String</strong> <strong>GetSyntax</strong> () const</td>
<td>get syntax as string</td>
</tr>
<tr>
<td>const <strong>ArgsBlock</strong> &amp; <strong>GetArguments</strong> () const</td>
<td>read-only access to command args</td>
</tr>
<tr>
<td><strong>ArgsBlock</strong> &amp; <strong>Arguments</strong> ()</td>
<td>read/write access to command arguments</td>
</tr>
<tr>
<td>const <strong>ArgsBlock</strong> &amp; <strong>GetResults</strong> () const</td>
<td>access to result arguments</td>
</tr>
<tr>
<td>const <strong>Util::String</strong> &amp; <strong>GetError</strong> () const</td>
<td>get error string</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>reevaluate class type</td>
</tr>
<tr>
<td>Function Name</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>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 __cdecl SetError (const char *fmt,...)
Member Function Documentation

void Scripting::Command::OnRegister() [virtual]
called when the scripting command is registered

Called when the command is registered with the script server. Derive this method and setup the arguments the command requires and the block of results the command returns.

void Scripting::Command::OnUnregister() [virtual]
called when the scripting command is unregistered

Called when the command is unregistered from the script server. This method must undo everything done in the OnRegister() method.

bool Scripting::Command::OnExecute() [virtual]
called when the script command is executed

Called when the script command should be executed. The command arguments will be filled with the args from the script function call, after the command functionality has been executed the command should provide any results in its results block. If there was an error executing the command the command should return false and have an error string set.

String Scripting::Command::GetSyntax() const
get syntax as string

Returns a string describing the command syntax in the form "type name = cmd(type name, type name, ...)"
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 Tue Feb 19 12:16:53 2008
Scripting::LuaServer
Scripting::LuaServer Class Reference

#include <luaserver.h>

Inheritance diagram for Scripting::LuaServer:
Detailed Description

LUA backend for the Nebula3 scripting subsystem.

(C) 2006 Radon Labs GmbH
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LuaServer()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual ~LuaServer()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual bool Open()</td>
<td>open the script server</td>
</tr>
<tr>
<td>virtual void Close()</td>
<td>close the script server</td>
</tr>
<tr>
<td>virtual void RegisterCommand(const Util::String &amp;name, const Ptr&lt; Command &gt; &amp;cmd)</td>
<td>register a command with the script server</td>
</tr>
<tr>
<td>virtual void UnregisterCommand(const Util::String &amp;name)</td>
<td>unregister a command from the script server</td>
</tr>
<tr>
<td>virtual bool Eval(const Util::String &amp;str)</td>
<td>evaluate a script statement in a string</td>
</tr>
<tr>
<td>bool IsOpen() const</td>
<td>return true if open</td>
</tr>
<tr>
<td>virtual bool EvalScript(const IO::URI &amp;uri)</td>
<td>evaluate a script file</td>
</tr>
<tr>
<td>bool HasCommand(const Util::String &amp;cmdName) const</td>
<td>return true if a command has been registered by name</td>
</tr>
<tr>
<td>SizeT GetNumCommands() const</td>
<td>return number of registered commands</td>
</tr>
<tr>
<td>const Ptr&lt; Command &gt; &amp; GetCommandByIndex(IndexT i)</td>
<td>return pointer to command at index</td>
</tr>
<tr>
<td>const Ptr&lt; Command &gt; &amp; GetCommandByName(const Util::String &amp;cmdName)</td>
<td>get pointer to command by command name</td>
</tr>
<tr>
<td>const Util::String &amp; GetError() const</td>
<td>get error string if evaluation fails</td>
</tr>
<tr>
<td>void PrintCommandList()</td>
<td></td>
</tr>
<tr>
<td>Function/Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>PrintCommandHelp(const Util::String &amp;cmdName) const</code></td>
<td>Print help on a command on stdout</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</th>
<th><strong>DumpRefCountingLeaks</strong> ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>dump refcounting leaks, call at end of application (NEBUA3_DEBUG builds only!)</em></td>
</tr>
</tbody>
</table>
**Protected Member Functions**

<table>
<thead>
<tr>
<th></th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>ClearError</strong> ()</td>
<td>clear current error</td>
</tr>
<tr>
<td>void</td>
<td><strong>setError</strong> (const Util::String &amp;err)</td>
<td>set error string</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
#include "Util/Types.h"

void Scripting::LuaServer::RegisterCommand(const Util::String name, const Ptr<Command>& cmd) [virtual]

register a command with the script server

Registers a new script command with the LUA server.

Reimplemented from Scripting::ScriptServer.

void Scripting::LuaServer::UnregisterCommand(const Util::String& name) [virtual]

unregister a command from the script server

Unregister a script command.

Reimplemented from Scripting::ScriptServer.

bool Scripting::LuaServer::Eval(const Util::String str) [virtual]

evaluate a script statement in a string

Evaluates a piece of LUA code in a string.

Reimplemented from Scripting::ScriptServer.

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.
Scripting::ScriptServer
Scripting::ScriptServer Class Reference

#include <scriptserver.h>

Inheritance diagram for Scripting::ScriptServer:

```
Core::RefCounted
      |     |
      v     v
Scripting::ScriptServer
      |     |
      v     v
Scripting::LuaServer
```
Detailed Description

Server class of the scripting subsystem. The scripting server keeps track of all registered class script interfaces and registered global script commands. Subclasses of script server know how to execute scripts of a specific language.

(C) 2006 Radon Labs
# Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ScriptServer</strong> ()</td>
<td>constructor</td>
</tr>
<tr>
<td>virtual <strong>~ScriptServer</strong> ()</td>
<td>destructor</td>
</tr>
<tr>
<td>virtual bool <strong>Open</strong> ()</td>
<td>open the script server</td>
</tr>
<tr>
<td>virtual void <strong>Close</strong> ()</td>
<td>close the script server</td>
</tr>
<tr>
<td>bool <strong>IsOpen</strong> () const</td>
<td>return true if open</td>
</tr>
<tr>
<td>virtual void <strong>RegisterCommand</strong> (const <strong>Util::String</strong> &amp;name, const <strong>Ptr&lt; Command &gt;</strong> &amp;cmd)</td>
<td>register a command with the script server</td>
</tr>
<tr>
<td>virtual void <strong>UnregisterCommand</strong> (const <strong>Util::String</strong> &amp;name)</td>
<td>unregister a command from the script server</td>
</tr>
<tr>
<td>virtual bool <strong>Eval</strong> (const <strong>Util::String</strong> &amp;str)</td>
<td>evaluate a script statement in a string</td>
</tr>
<tr>
<td>virtual bool <strong>EvalScript</strong> (const <strong>IO::URI</strong> &amp;uri)</td>
<td>evaluate a script file</td>
</tr>
<tr>
<td>bool <strong>HasCommand</strong> (const <strong>Util::String</strong> &amp;cmdName) const</td>
<td>return true if a command has been registered by name</td>
</tr>
<tr>
<td><strong>SizeT</strong> <strong>GetNumCommands</strong> () const</td>
<td>return number of registered commands</td>
</tr>
<tr>
<td><strong>const <strong>Ptr&lt; Command &gt;</strong> &amp;</strong> GetCommandByIndex** (IndexT i) const</td>
<td>return pointer to command at index</td>
</tr>
<tr>
<td><strong>const <strong>Ptr&lt; Command &gt;</strong> &amp;</strong> GetCommandByName** (const <strong>Util::String</strong> &amp;cmdName) const</td>
<td>get pointer to command by command name</td>
</tr>
<tr>
<td><strong>const <strong>Util::String</strong> &amp;</strong> GetError** () const</td>
<td>get error string if evaluation failes</td>
</tr>
<tr>
<td><strong>void</strong> <strong>PrintCommandList</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>PrintCommandHelp (const Util::String &amp;cmdName) const</code></td>
<td>print all registered commands to stdout</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>GetRefCount () const</code></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><code>int GetRefCount () const</code></td>
<td>get the current refcount</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>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void ClearError ()</code></td>
<td>clear current error</td>
</tr>
<tr>
<td><code>void SetError (const Util::String &amp;err)</code></td>
<td>set error string</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.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

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 CmdLineArgs& 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
# Static Public Member Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static short</td>
<td><strong>ConvertShort</strong></td>
<td>(Type fromByteOrder, Type toByteOrder, short val)</td>
</tr>
<tr>
<td></td>
<td><strong>endian-convert short value</strong></td>
<td></td>
</tr>
<tr>
<td>static ushort</td>
<td><strong>ConvertUShort</strong></td>
<td>(Type fromByteOrder, Type toByteOrder, ushort val)</td>
</tr>
<tr>
<td></td>
<td><strong>endian-convert ushort value</strong></td>
<td></td>
</tr>
<tr>
<td>static int</td>
<td><strong>ConvertInt</strong></td>
<td>(Type fromByteOrder, Type toByteOrder, int val)</td>
</tr>
<tr>
<td></td>
<td><strong>endian-convert int value</strong></td>
<td></td>
</tr>
<tr>
<td>static uint</td>
<td><strong>ConvertUInt</strong></td>
<td>(Type fromByteOrder, Type toByteOrder, uint val)</td>
</tr>
<tr>
<td></td>
<td><strong>endian-convert uint value</strong></td>
<td></td>
</tr>
<tr>
<td>static float</td>
<td><strong>ConvertFloat</strong></td>
<td>(Type fromByteOrder, Type toByteOrder, float val)</td>
</tr>
<tr>
<td></td>
<td><strong>endian-convert float value</strong></td>
<td></td>
</tr>
<tr>
<td>static double</td>
<td><strong>ConvertDouble</strong></td>
<td>(Type fromByteOrder, Type toByteOrder, double val)</td>
</tr>
<tr>
<td></td>
<td><strong>endian-convert double value</strong></td>
<td></td>
</tr>
<tr>
<td>static Math::float4</td>
<td><strong>ConvertFloat4</strong></td>
<td>(Type fromByteOrder, Type toByteOrder, const Math::float4 &amp;val)</td>
</tr>
<tr>
<td></td>
<td><strong>endian-convert float4 value</strong></td>
<td></td>
</tr>
<tr>
<td>static Math::matrix44</td>
<td><strong>ConvertMatrix44</strong></td>
<td>(Type fromByteOrder, Type toByteOrder, const Math::matrix44 &amp;val)</td>
</tr>
<tr>
<td></td>
<td><strong>endian convert matrix44 value</strong></td>
<td></td>
</tr>
</tbody>
</table>
System::Cpu
System::Cpu Class Reference

#include <cpu.h>
Detailed Description

Provides information about the system's CPU(s).

(C) 2007 Radon Labs GmbH
Public Types

enum CoreId
core id's
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 enumeration</td>
<td></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>static bool</td>
<td><strong>Exists</strong> <em>(RootKey rootKey, const Util::String &amp;key, const Util::String &amp;name)</em></td>
<td>return true if a registry entry exists</td>
</tr>
<tr>
<td>static bool</td>
<td><strong>Write</strong> <em>(RootKey rootKey, const Util::String &amp;key, const Util::String &amp;name, const Util::String &amp;value)</em></td>
<td>write a registry entry</td>
</tr>
<tr>
<td>static Util::String</td>
<td><strong>Read</strong> <em>(RootKey rootKey, const Util::String &amp;key, const Util::String &amp;name)</em></td>
<td>read a registry entry</td>
</tr>
<tr>
<td>static bool</td>
<td><strong>Delete</strong> <em>(RootKey rootKey, const Util::String &amp;key)</em></td>
<td>delete a registry key (and all its contained values)</td>
</tr>
</tbody>
</table>
Member Function Documentation

bool System::Win32Registry::Exists( 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.

bool System::Win32Registry::Write( 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.

Util::String System::Win32Registry::Read( RootKey rootKey, 
const Util::String key, 
& 
const Util::String name 
& 
)
read a registry entry

Get a value from the registry. Fails hard if the key doesn't exists (use the \texttt{Exists()} method to make sure that the key exists!).

\begin{verbatim}
bool System::Win32Registry::Delete( RootKey rootKey,
    const Util::String & key &
) [static]
\end{verbatim}

delete a registry key (and all its contained values)

This deletes a complete registry key with all its values.
Threading::Barrier
#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:

```
Win32::Win32CriticalSection

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
Public Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>Enter ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enter the critical section</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>Leave ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>leave the critical section</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:53 2008
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>void Signal ()</td>
<td>signal the event</td>
</tr>
<tr>
<td>void Wait () const</td>
<td>wait for the event to become signalled</td>
</tr>
<tr>
<td>bool WaitTimeout (int ms) const</td>
<td>wait for the event with timeout in millisecs</td>
</tr>
<tr>
<td>bool Peek () const</td>
<td>check if event is signalled</td>
</tr>
</tbody>
</table>
**Member Function Documentation**

```cpp
bool Win32::Win32Event::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.

```cpp
bool Win32::Win32Event::Peek() const [inline, inherited]
```

check if event is signalled

This checks if the event is signalled and returns immediately.
Threading::Interlocked
#include <interlocked.h>

Inheritance diagram for Threading::Interlocked:
Detailed Description

Provide simple atomic operations on memory variables.

(C) 2006 Radon Labs GmbH
Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Increment</strong></td>
<td>(int volatile &amp;var)</td>
</tr>
<tr>
<td><strong>Decrement</strong></td>
<td>(int volatile &amp;var)</td>
</tr>
<tr>
<td><strong>Add</strong></td>
<td>(int volatile &amp;var, int add)</td>
</tr>
</tbody>
</table>
Threading::SafePriorityQueue
#include <safepriorityqueue.h>

Inheritance diagram for Threading::SafePriorityQueue< PRITYPE, TYPE >:

```
Util::Queue< Util::KeyValuePair< PRITYPE, TYPE > >
```
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><strong>constructor</strong></td>
</tr>
<tr>
<td><strong>SafePriorityQueue</strong> (const SafePriorityQueue&lt; PRITYPE, TYPE &gt; &amp;rhs)</td>
<td><strong>copy constructor</strong></td>
</tr>
<tr>
<td>void <strong>operator=</strong> (const SafePriorityQueue&lt; PRITYPE, TYPE &gt; &amp;rhs)</td>
<td><strong>assignment operator</strong></td>
</tr>
<tr>
<td>SizeT <strong>Size ()</strong> const</td>
<td>returns number of elements in the queue</td>
</tr>
<tr>
<td>bool <strong>IsEmpty ()</strong> const</td>
<td>return true if queue is empty</td>
</tr>
<tr>
<td>void <strong>Clear ()</strong></td>
<td>remove all elements from the queue</td>
</tr>
<tr>
<td>void <strong>Insert</strong> (PRITYPE pri, const TYPE &amp;e)</td>
<td>add element to the back of the queue</td>
</tr>
<tr>
<td>void <strong>EraseMatchingElements</strong> (const TYPE &amp;e)</td>
<td>erase all matching elements</td>
</tr>
<tr>
<td>TYPE <strong>Dequeure</strong> ()</td>
<td>remove the element from the front of the queue</td>
</tr>
<tr>
<td>TYPE <strong>Peek</strong> () const</td>
<td>get copy of element at front of queue without removing it</td>
</tr>
<tr>
<td>void <strong>Wait</strong> ()</td>
<td>wait until queue contains at least one element</td>
</tr>
<tr>
<td>void <strong>Signal</strong> ()</td>
<td>signal the internal event, so that <strong>Wait()</strong> will return</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Util::KeyValuePair</strong>&lt;PRITYPE, TYPE&gt; &amp; <strong>operator[]</strong> (IndexT index) const</td>
<td>access element by index, 0 is the frontmost element (next to be dequeued)</td>
</tr>
<tr>
<td>bool <strong>operator==</strong> (const Queue&lt;Util::KeyValuePair&lt;PRITYPE, TYPE&gt;&gt; &amp;rhs) const</td>
<td>equality operator</td>
</tr>
<tr>
<td>bool <strong>operator!=</strong> (const Queue&lt;Util::KeyValuePair&lt;PRITYPE, TYPE&gt;&gt; &amp;rhs) const</td>
<td>inequality operator</td>
</tr>
<tr>
<td>bool <strong>Contains</strong> (const Util::KeyValuePair&lt;PRITYPE, TYPE&gt; &amp;e) const</td>
<td>return true if queue contains element</td>
</tr>
<tr>
<td>void <strong>Enqueue</strong> (const Util::KeyValuePair&lt;PRITYPE, TYPE&gt; &amp;e)</td>
<td>add element to the back of the queue</td>
</tr>
</tbody>
</table>
signal the internal event, so that \texttt{Wait()} will return

This signals the internal event object, on which \texttt{Wait()} may be waiting. This method may be useful to wake up a thread waiting for events when it should stop.
Threading::SafeQueue
Threading::SafeQueue< TYPE > Class Template Reference

#include <safequeue.h>

Inheritance diagram for Threading::SafeQueue< TYPE >:
Detailed Description

template<class TYPE>
class Threading::SafeQueue< TYPE >

Thread-safe version of Util::Queue.

(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>void <strong>operator= (const SafeQueue&lt; TYPE &gt; &amp;rhs)</strong></td>
<td>assignment operator</td>
</tr>
<tr>
<td>SizeT <strong>Size ()</strong></td>
<td>returns number of elements in the queue</td>
</tr>
<tr>
<td>bool <strong>IsEmpty ()</strong></td>
<td>return true if queue is empty</td>
</tr>
<tr>
<td>void <strong>Clear ()</strong></td>
<td>remove all elements from the queue</td>
</tr>
<tr>
<td>void <strong>Enqueue (const TYPE &amp;e)</strong></td>
<td>add element to the back of the queue</td>
</tr>
<tr>
<td>TYPE <strong>Dequeue ()</strong></td>
<td>remove the element from the front of the queue</td>
</tr>
<tr>
<td>Util::Array&lt; TYPE &gt; <strong>DequeueAll ()</strong></td>
<td>dequeue all events (only requires one lock)</td>
</tr>
<tr>
<td>TYPE <strong>Peek ()</strong></td>
<td>access to element at front of queue without removing it</td>
</tr>
<tr>
<td>void <strong>Wait ()</strong></td>
<td>wait until queue contains at least one element</td>
</tr>
<tr>
<td>void <strong>Signal ()</strong></td>
<td>signal the internal event, so that Wait() will return</td>
</tr>
<tr>
<td>void <strong>EraseMatchingElements (const TYPE &amp;e)</strong></td>
<td>erase all matching elements</td>
</tr>
</tbody>
</table>
### Protected Member Functions

<table>
<thead>
<tr>
<th>Type &amp;</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE &amp;</td>
<td><code>operator[]</code> (IndexT index) const</td>
<td>Access element by index, 0 is the frontmost element (next to be dequeued)</td>
</tr>
<tr>
<td>bool</td>
<td><code>operator==</code> (const Queue&lt; TYPE &gt; &amp;rhs) const</td>
<td>Equality operator</td>
</tr>
<tr>
<td>bool</td>
<td><code>operator!=</code> (const Queue&lt; TYPE &gt; &amp;rhs) const</td>
<td>Inequality operator</td>
</tr>
<tr>
<td>bool</td>
<td><code>Contains</code> (const TYPE &amp;e) const</td>
<td>Return true if queue contains element</td>
</tr>
</tbody>
</table>
template<class TYPE>
void Threading::SafeQueue< TYPE >::Signal

signal the internal event, so that \texttt{Wait()} will return

This signals the internal event object, on which \texttt{Wait()} may be waiting. This method may be useful to wake up a thread waiting for events when it should stop.
Threading::Thread
Threading::Thread Class Reference

#include <thread.h>

Inheritance diagram for Threading::Thread:
Detailed Description

Todo:
   describe Thread class

(C) 2006 Radon Labs GmbH
Public Types

enum Priority

thread priorities
## Public Member Functions

<table>
<thead>
<tr>
<th>function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<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 (unsigned int s)</td>
<td>set stack size in bytes (default is 4 KByte)</td>
</tr>
<tr>
<td>unsigned int GetStackSize () const</td>
<td>get stack size</td>
</tr>
<tr>
<td>voidSetName (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 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>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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static const char * GetMyThreadName()</code></td>
<td>obtain name of thread from within thread code</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>virtual void EmitWakeupSignal()</td>
<td>override this method if your thread loop needs a wakeup call before stopping</td>
</tr>
<tr>
<td>virtual void DoWork()</td>
<td>this method runs in the thread context</td>
</tr>
<tr>
<td>bool ThreadStopRequested() const</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 Win32::Win32Thread::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 & Win32::Win32Thread::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 Win32::Win32Thread::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 returns immediately without waiting for the thread to start.

```cpp
void Win32::Win32Thread::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.

    const char *
Win32::Win32Thread::GetMyThreadName ( ) [static, inherited]

obtain name of thread from within thread code

Static method to obtain the current thread name from anywhere in the thread's code.

    void
Win32::Win32Thread::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.

    void
Win32::Win32Thread::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.

    bool
Win32::Win32Thread::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::ThreadLocalPtr
Threading::ThreadLocalPtr< T > Class Template Reference

#include <threadlocalptr.h>
Detailed Description

template<typename T>
class Threading::ThreadLocalPtr< T >

Todo:
describe ThreadLocalPtr class

(C) 2006 Radon Labs GmbH
Timing::CalendarTime
Timing::CalendarTime Class Reference

#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
# Public Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Declaration</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enum</code></td>
<td><code>Month</code></td>
</tr>
<tr>
<td></td>
<td><code>months enum</code></td>
</tr>
<tr>
<td><code>enum</code></td>
<td><code>Weekday</code></td>
</tr>
<tr>
<td></td>
<td><code>weekdays enum</code></td>
</tr>
<tr>
<td><code>typedef</code></td>
<td><code>unsigned int Year</code></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><strong>void SetYear (Year y)</strong></td>
<td>set the year</td>
</tr>
<tr>
<td><strong>Year GetYear () const</strong></td>
<td>get the year</td>
</tr>
<tr>
<td><strong>void SetMonth (Month m)</strong></td>
<td>set the month</td>
</tr>
<tr>
<td><strong>Month GetMonth () const</strong></td>
<td>get the month</td>
</tr>
<tr>
<td><strong>void SetWeekday (Weekday wd)</strong></td>
<td>set the day-of-week</td>
</tr>
<tr>
<td><strong>Weekday GetWeekday () const</strong></td>
<td>get the day-of-week</td>
</tr>
<tr>
<td><strong>void SetDay (Day d)</strong></td>
<td>set the day (of month)</td>
</tr>
<tr>
<td><strong>Day GetDay () const</strong></td>
<td>get the day (of month)</td>
</tr>
<tr>
<td><strong>void SetHour (Hour h)</strong></td>
<td>set hour-of-day</td>
</tr>
<tr>
<td><strong>Hour GetHour () const</strong></td>
<td>get hour-of-day</td>
</tr>
<tr>
<td><strong>void SetMinute (Minute m)</strong></td>
<td>set minute-of-hour</td>
</tr>
<tr>
<td><strong>Minute GetMinute () const</strong></td>
<td>get minute-of-hour</td>
</tr>
<tr>
<td><strong>void SetSecond (Second s)</strong></td>
<td>set second-of-minute</td>
</tr>
<tr>
<td><strong>Second GetSecond () const</strong></td>
<td>get second-of-minute</td>
</tr>
<tr>
<td><strong>void SetMilliSecond (MilliSecond ms)</strong></td>
<td>set milliseconds</td>
</tr>
<tr>
<td><strong>MilliSecond GetMilliSecond () const</strong></td>
<td>get milliseconds</td>
</tr>
</tbody>
</table>
## Static Public Member Functions

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

**CalendarTime**  
Win32::Win32CalendarTime::GetSystemTime( ) [static, inherited]

get the current system time

Obtains the current system time. This does not depend on the current time zone.

Reimplemented from **Base::CalendarTimeBase**.

**CalendarTime**  
Win32::Win32CalendarTime::GetLocalTime( ) [static, inherited]

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
Timing::InputTimeSource
# Timing::InputTimeSource Class Reference

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

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:53 2008
Timing::SystemTimeSource
# Timing::SystemTimeSource Class Reference

#include <systemtimesource.h>
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:

```cpp
Time sysTime = SystemTimeSource::Instance()->GetTime();
```

(C) 2007 Radon Labs GmbH
Timing::TimeManager
Timing::TimeManager Class Reference

#include <timemanager.h>
Detailed Description

Singleton object which manages the current game time. These are the standard time source objects provided by Mangalore:

**SystemTimeSource** - timing for low level Mangalore subsystems
**GameTimeSource** - timing for the game logic
**CameraTimeSource** - extra time source for camera handling
**GuiTimeSource** - time source for user interface stuff

(C) 2005 Radon Labs GmbH
• Main Page
• Namespaces
• Data Structures
• Files
• Related Pages

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

**Timing::Timer**
Timing::Timer Class Reference

#include <timer.h>

Inheritance diagram for Timing::Timer:

```
Win32::Win32Timer

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
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start</strong> ()</td>
<td>start/continue the timer</td>
</tr>
<tr>
<td><strong>Stop</strong> ()</td>
<td>stop the timer</td>
</tr>
<tr>
<td><strong>Reset</strong> ()</td>
<td>reset the timer</td>
</tr>
<tr>
<td><strong>Running</strong> () const</td>
<td>return true if currently running</td>
</tr>
<tr>
<td><strong>GetTime</strong> () const</td>
<td>get current time in seconds</td>
</tr>
<tr>
<td><strong>GetTicks</strong> () const</td>
<td>get current time in ticks</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Win32::Win32Timer::Start() [inherited]
```

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).

```cpp
void Win32::Win32Timer::Stop() [inherited]
```

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.

```cpp
void Win32::Win32Timer::Reset() [inherited]
```

reset the timer

Reset the timer so that will start counting at zero again.

```cpp
bool Win32::Win32Timer::Running() const [inherited]
```

return true if currently running

Returns true if the timer is currently running.

```cpp
Timing::Time Win32::Win32Timer::GetTime() const [inherited]
```

get current time in seconds
This returns the timer's current time in seconds.

```cpp
uint Win32::Win32Timer::GetTicks() const [inherited]
```

get current time in ticks

This returns the timer's current time in "ticks".

The Nebula Device 3 documentation generated by `doxygen` at Tue Feb 19 12:16:53 2008
• Main Page
• Namespaces
• Data Structures
• Files
• Related Pages

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

Timing::TimeSource
Timing::TimeSource Class Reference

#include <timesource.h>
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 unpause independently from each other, and they may also run faster or slower then 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.:

```cpp
Time gameTime = GameTime::Instance()->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 timesources. No time source should set the time in a subsystem activly.

(C) 2007 Radon Labs GmbH
Timing::TimingTimeSource
Timing::TimingTimeSource Class Reference

#include <gametimesource.h>
Detailed Description

Provides timing information for the **Timing** logic.

(C) 2007 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

typedef TYPE * Iterator

declare iterator
**Public Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Array ()</code></td>
<td>constructor with default parameters</td>
</tr>
<tr>
<td><code>Array (SizeT initialCapacity, SizeT initialGrow)</code></td>
<td>constructor with initial size and grow size</td>
</tr>
<tr>
<td><code>Array (SizeT initialSize, SizeT initialGrow, const TYPE &amp;initialValue)</code></td>
<td>constructor with initial size, grow size and initial values</td>
</tr>
<tr>
<td><code>Array (const Array&lt;TYPE&gt; &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>~Array ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void operator= (const Array&lt;TYPE&gt; &amp;rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td><code>TYPE &amp; operator[] (IndexT index) const</code></td>
<td>index operator</td>
</tr>
<tr>
<td><code>bool operator==(const Array&lt;TYPE&gt; &amp;rhs) const</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const Array&lt;TYPE&gt; &amp;rhs) const</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>void Append (const TYPE &amp;elm)</code></td>
<td>append element to end of array</td>
</tr>
<tr>
<td><code>void AppendArray (const Array&lt;TYPE&gt; &amp;rhs)</code></td>
<td>append the contents of an array to this array</td>
</tr>
<tr>
<td><code>void Reserve (SizeT num)</code></td>
<td>reserve num elements at end of array</td>
</tr>
<tr>
<td><code>SizeT Size () const</code></td>
<td>get number of elements in array</td>
</tr>
<tr>
<td><code>SizeT Capacity () const</code></td>
<td>get overall allocated size of array in number of elements</td>
</tr>
<tr>
<td><code>void Set (IndexT index, const TYPE &amp;elm)</code></td>
<td>set element at index, grow array if necessary</td>
</tr>
<tr>
<td><code>TYPE &amp; At (IndexT index)</code></td>
<td>return reference to nth element in array</td>
</tr>
<tr>
<td>TYPE &amp;</td>
<td>Front () const</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>return reference to first element</td>
</tr>
<tr>
<td>TYPE &amp;</td>
<td>Back () const</td>
</tr>
<tr>
<td></td>
<td>return reference to last element</td>
</tr>
<tr>
<td>bool</td>
<td>IsEmpty () const</td>
</tr>
<tr>
<td></td>
<td>return true if array empty</td>
</tr>
<tr>
<td>void</td>
<td>EraseIndex (IndexT index)</td>
</tr>
<tr>
<td></td>
<td>erase element at index</td>
</tr>
<tr>
<td>Iterator</td>
<td>Erase (Iterator iter)</td>
</tr>
<tr>
<td></td>
<td>erase element pointed to by iterator</td>
</tr>
<tr>
<td>void</td>
<td>Insert (IndexT index, const TYPE &amp;elm)</td>
</tr>
<tr>
<td></td>
<td>insert element before element at index</td>
</tr>
<tr>
<td>void</td>
<td>InsertSorted (const TYPE &amp;elm)</td>
</tr>
<tr>
<td></td>
<td>insert element into sorted array</td>
</tr>
<tr>
<td>void</td>
<td>Clear ()</td>
</tr>
<tr>
<td></td>
<td>clear array (calls destructors)</td>
</tr>
<tr>
<td>void</td>
<td>Reset ()</td>
</tr>
<tr>
<td></td>
<td>reset array (does NOT call destructors)</td>
</tr>
<tr>
<td>Iterator</td>
<td>Begin () const</td>
</tr>
<tr>
<td></td>
<td>return iterator to beginning of array</td>
</tr>
<tr>
<td>Iterator</td>
<td>End () const</td>
</tr>
<tr>
<td></td>
<td>return iterator to end of array</td>
</tr>
<tr>
<td>Iterator</td>
<td>Find (const TYPE &amp;elm) const</td>
</tr>
<tr>
<td></td>
<td>find identical element in array, return iterator</td>
</tr>
<tr>
<td>IndexT</td>
<td>FindIndex (const TYPE &amp;elm) const</td>
</tr>
<tr>
<td></td>
<td>find identical element in array, return index, InvalidIndex if not found</td>
</tr>
<tr>
<td>void</td>
<td>Fill (IndexT first, SizeT num, const TYPE &amp;elm)</td>
</tr>
<tr>
<td></td>
<td>fill array range with element</td>
</tr>
<tr>
<td>void</td>
<td>Reallocate (SizeT capacity, SizeT grow)</td>
</tr>
<tr>
<td></td>
<td>clear contents and preallocate with new attributes</td>
</tr>
<tr>
<td>Array&lt; TYPE &gt;</td>
<td>Difference (const Array&lt; TYPE &gt; &amp;rhs)</td>
</tr>
<tr>
<td></td>
<td>returns new array with elements which are not in rhs (slow!)</td>
</tr>
<tr>
<td>void</td>
<td>Sort ()</td>
</tr>
<tr>
<td></td>
<td>sort the array</td>
</tr>
<tr>
<td>IndexT</td>
<td>BinarySearchIndex (const TYPE &amp;elm) const</td>
</tr>
<tr>
<td></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 [inline] >::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 [inline] >::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 [inline] >::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) [inline] >::Reserve

reserve num elements at end of array
Make room for N new elements at the end of the array, and return a pointer to the start of the reserved area. This can be (carefully!) used as a fast shortcut to fill the array directly with data.

```cpp
template<class TYPE>
TYPE &
Util::Array< (IndexT index ) [inline]
TYPE >::At

return reference to nth element in array

Access an element. This method may grow the array if the index is outside the array range.

```cpp
template<class TYPE>
void
Util::Array< const
TYPE &
&
extended
>
::InsertSorted

insert element into sorted array

This inserts the element into a sorted array. In the current implementation this is a slow operation O(n). This should be optimized to O(log n).

```cpp
template<class TYPE>
void
Util::Array< ( ) [inline]
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.

```cpp
template<class TYPE>
void
Util::Array< ( ) [inline]
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!

```cpp
template<class TYPE>
Array<
TYPE
>:::Iterator
Util::Array<
TYPE
>:::Find
(const TYPE& elm) const [inline]
```

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) const [inline]
```

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<
IndexT
first,
```
TYPE >::Fill
    SizeT num,
    const TYPE & elm
    ) [inline]

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
    num    num elements to fill
    elm    fill value

template<class TYPE>
    Array< TYPE >
    Util::Array< ( TYPE rhs ) [inline]
    >::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< ( ) [inline]
    TYPE >::Sort

sort the array

Sorts the array. This just calls the STL sort algorithm.
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::Atom
Util::Atom< TYPE > Class Template Reference

#include <atom.h>


Detailed Description

template<class TYPE>
class Util::Atom< TYPE >

An Atom is a compact, shared reference of a constant object. A unique object is guaranteed to exist only once, no matter how many Atoms are pointing to it. Copying Atoms and comparing against Atoms is very fast, since these are actually pointer operations.

Note that comparing 2 Atoms against each other will yield a different result then comparing the contents of the same Atoms. This is because comparing 2 Atoms actually compares the addresses of the content without actually invoking the compare operators of the content. So, comparing 2 Atom<String>'s will never do a strcmp(), instead the string pointers are compared against each other.

Atoms are consistent across threads.

The Atom table will *not* be garbage collected automatically, this means, entries for which no more Atoms exist will not be removed automatically. This is to prevent excessive allocations/deallocations for some usage scenarios. Instead, if you want to remove orphaned entries, call the PerformGarbageCollection() method manually.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Atom</strong> ()</td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><strong>Atom</strong> (const TYPE &amp;rhs)</td>
<td><em>construct from type</em></td>
</tr>
<tr>
<td><strong>Atom</strong> (const <strong>Atom</strong>&lt; TYPE &gt; &amp;rhs)</td>
<td><em>copy constructor</em></td>
</tr>
<tr>
<td>~<strong>Atom</strong> ()</td>
<td><em>destructor</em></td>
</tr>
<tr>
<td>void <strong>operator</strong>= (const TYPE &amp;rhs)</td>
<td><em>assignment from type</em></td>
</tr>
<tr>
<td>void <strong>operator</strong>= (const <strong>Atom</strong>&lt; TYPE &gt; &amp;rhs)</td>
<td><em>assignment from other atom</em></td>
</tr>
<tr>
<td>bool <strong>operator</strong>== (const TYPE &amp;rhs) const</td>
<td><em>equality operator with atom content</em></td>
</tr>
<tr>
<td>bool <strong>operator</strong>!= (const TYPE &amp;rhs) const</td>
<td><em>inequality operator with atom content</em></td>
</tr>
<tr>
<td>bool <strong>operator</strong>&gt; (const TYPE &amp;rhs) const</td>
<td><em>greater-then operator with atom content</em></td>
</tr>
<tr>
<td>bool <strong>operator</strong>&lt; (const TYPE &amp;rhs) const</td>
<td><em>less-then operator with atom content</em></td>
</tr>
<tr>
<td>bool <strong>operator</strong>&gt;= (const TYPE &amp;rhs) const</td>
<td><em>greater-or-equal operator with atom content</em></td>
</tr>
<tr>
<td>bool <strong>operator</strong>&lt;= (const TYPE &amp;rhs) const</td>
<td><em>less-or-equal operator with atom content</em></td>
</tr>
<tr>
<td>bool <strong>operator</strong>== (const <strong>Atom</strong>&lt; TYPE &gt; &amp;rhs) const</td>
<td><em>equality operator between atoms</em></td>
</tr>
<tr>
<td>bool <strong>operator</strong>!= (const <strong>Atom</strong>&lt; TYPE &gt; &amp;rhs) const</td>
<td><em>inequality operator between atoms</em></td>
</tr>
<tr>
<td>bool <strong>operator</strong>&gt; (const <strong>Atom</strong>&lt; TYPE &gt; &amp;rhs) const</td>
<td><em>greater-then operator between atoms</em></td>
</tr>
<tr>
<td>bool <strong>operator</strong>&lt; (const <strong>Atom</strong>&lt; TYPE &gt; &amp;rhs) const</td>
<td><em>less-then operator between atoms</em></td>
</tr>
<tr>
<td>bool <strong>operator</strong>&gt;= (const <strong>Atom</strong>&lt; TYPE &gt; &amp;rhs) const</td>
<td><em>greater-or-equal operator between atoms</em></td>
</tr>
</tbody>
</table>
greater-or-equal operator between atoms

bool operator<= (const Atom< TYPE > &rhs) const

less-or-equal operator between atoms

void Clear ()

clear content

bool IsValid () const

return true if the Atom is valid

const TYPE & Value () const

access to referenced object
## 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>PerformGarbageCollection</strong> ()</td>
<td>perform garbage collection on atom table</td>
</tr>
<tr>
<td>static SizeT</td>
<td><strong>GetAtomTableSize</strong> ()</td>
<td>get current atom table size (for debugging)</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void * operator new (size_t size)</code></td>
<td>overloaded operator new</td>
</tr>
<tr>
<td><code>void operator delete (void *p)</code></td>
<td>overloaded operator delete</td>
</tr>
<tr>
<td><code>Blob ()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>Blob (const void *ptr, SizeT size)</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>Blob (SizeT size)</code></td>
<td>reserve N bytes</td>
</tr>
<tr>
<td><code>Blob (const Blob &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>~Blob ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void operator= (const Blob &amp;rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td><code>bool operator== (const Blob &amp;rhs)</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>bool operator!= (const Blob &amp;rhs)</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>bool operator&gt; (const Blob &amp;rhs)</code></td>
<td>greater operator</td>
</tr>
<tr>
<td><code>bool operator&lt; (const Blob &amp;rhs)</code></td>
<td>less operator</td>
</tr>
<tr>
<td><code>bool operator&gt;= (const Blob &amp;rhs)</code></td>
<td>greater-equal operator</td>
</tr>
<tr>
<td><code>bool operator&lt;= (const Blob &amp;rhs)</code></td>
<td>less-equal operator</td>
</tr>
<tr>
<td><code>bool IsValid ()</code></td>
<td>return true if the blob contains data</td>
</tr>
<tr>
<td><code>void Reserve (SizeT size)</code></td>
<td>reserve N bytes</td>
</tr>
<tr>
<td><code>void Set (const void *ptr, SizeT size)</code></td>
<td></td>
</tr>
<tr>
<td>Set blob contents</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>void * <strong>GetPtr () const</strong></td>
<td></td>
</tr>
<tr>
<td>get blob ptr</td>
<td></td>
</tr>
<tr>
<td><strong>SizeT Size () const</strong></td>
<td></td>
</tr>
<tr>
<td>get blob size</td>
<td></td>
</tr>
<tr>
<td><strong>IndexT HashCode () const</strong></td>
<td></td>
</tr>
<tr>
<td>get a hash code (compatible with <em>Util::HashTable</em>)</td>
<td></td>
</tr>
</tbody>
</table>
Static Public Member Functions

```
static void Setup ()
static Setup method, called by Util::Setup()
```

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:53 2008
Util::CharEnhancementUtil
Util::CharEnhancementUtil Class Reference

#include <charenhancementutil.h>
Detailed Description

Manage graphics character entity attachments and layered skins.

(C) 2006 Radon Labs GmbH
Util::CommandLineArgs
Util::CmdLineArgs Class Reference

#include <cmdlineargs.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>CmdLineArgs()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>CmdLineArgs(const String &amp;cmdLine)</code></td>
<td>construct from Win32-style command string</td>
</tr>
<tr>
<td><code>CmdLineArgs(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)</code></td>
<td>get bool value</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 argument at index</td>
</tr>
<tr>
<td><code>int GetIntAtIndex(IndexT index)</code></td>
<td>get int value at index</td>
</tr>
<tr>
<td>Type</td>
<td>Function Name</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>float</td>
<td><code>GetFloatAtIndex</code> (IndexT index) const</td>
</tr>
<tr>
<td>bool</td>
<td><code>GetBoolAtIndex</code> (IndexT index) const</td>
</tr>
<tr>
<td>Math::float4</td>
<td><code>GetFloat4AtIndex</code> (IndexT index) const</td>
</tr>
<tr>
<td>Math::matrix44</td>
<td><code>GetMatrix44AtIndex</code> (IndexT index) const</td>
</tr>
</tbody>
</table>
Member Function Documentation

const String &
Util::CmdLineArgs::GetCmdName() const

getc the command name

Returns the command name.

bool
Util::CmdLineArgs::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></td>
<td>(unsigned char *buf, unsigned int numBytes) continue computing checksum</td>
</tr>
<tr>
<td><strong>End ()</strong></td>
<td>finish computing the checksum</td>
</tr>
<tr>
<td><strong>GetResult ()</strong></td>
<td>get result</td>
</tr>
<tr>
<td><strong>GetResult ()</strong></td>
<td>const</td>
</tr>
</tbody>
</table>
Member Function Documentation

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::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 can be added at any time with the \texttt{Add()} methods. Internally, lazy evaluation is used to keep the array sorted only when it needs to be sorted. Thus adding many keys is very fast, but the first search may be slow because that's when the sorting happens.

(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</strong> (const Dictionary&lt; KEYTYPE, VALUETYPE &gt; &amp;rhs)</td>
<td>copy constructor</td>
</tr>
<tr>
<td>void operator= (const Dictionary&lt; KEYTYPE, VALUETYPE &gt; &amp;rhs)</td>
<td>assignment operator</td>
</tr>
<tr>
<td>VALUETYPE &amp; operator[] (const KEYTYPE &amp;key)</td>
<td>read/write [] operator</td>
</tr>
<tr>
<td>const VALUETYPE &amp; operator[] (const KEYTYPE &amp;key) const</td>
<td>read-only [] operator</td>
</tr>
<tr>
<td>SizeT Size () const</td>
<td>return number of key/value pairs in the dictionary</td>
</tr>
<tr>
<td>void Clear ()</td>
<td>clear the dictionary</td>
</tr>
<tr>
<td>bool IsEmpty () const</td>
<td>return true if empty</td>
</tr>
<tr>
<td>void Add (const KeyValuePair&lt; KEYTYPE, VALUETYPE &gt; &amp;kvp)</td>
<td>add a key/value pair</td>
</tr>
<tr>
<td>void Add (const KEYTYPE &amp;key, const VALUETYPE &amp;value)</td>
<td>add a key and associated value</td>
</tr>
<tr>
<td>void Erase (const KEYTYPE &amp;key)</td>
<td>erase a key and its associated value</td>
</tr>
<tr>
<td>void EraseAtIndex (IndexT index)</td>
<td>erase a key at index</td>
</tr>
<tr>
<td>IndexT FindIndex (const KEYTYPE &amp;key) const</td>
<td>find index of key/value pair (InvalidIndex if doesn't exist)</td>
</tr>
<tr>
<td>bool Contains (const KEYTYPE &amp;key) const</td>
<td>return true if key exists in the array</td>
</tr>
<tr>
<td>const KEYTYPE &amp; KeyAtIndex (IndexT index) const</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>getValueAtIndex(IndexT index)</code></td>
<td>get a value at given index</td>
</tr>
<tr>
<td><code>KeyValuePairAtIndex(IndexT index) const</code></td>
<td>get key/value pair at index</td>
</tr>
<tr>
<td><code>KeysAsArray()</code> const</td>
<td>get all keys as array (slow)</td>
</tr>
<tr>
<td><code>ValuesAsArray()</code> const</td>
<td>get all values as array (slow)</td>
</tr>
</tbody>
</table>
Protected Member Functions

```cpp
void SortIfDirty () const
make sure the key value pair array is sorted
```
Util::FixedArray
Util::FixedArray< TYPE > Class
Template Reference

#include <fixedarray.h>
Template 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) const</code></td>
<td>write [] operator</td>
</tr>
<tr>
<td><code>bool operator== (const FixedArray&lt; TYPE &gt; &amp;rhs) const</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>bool operator!=(const FixedArray&lt; TYPE &gt; &amp;rhs) const</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 () const</code></td>
<td>get number of elements</td>
</tr>
<tr>
<td><code>void Clear (const TYPE &amp;val)</code></td>
<td>clear the array with 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 () const</code></td>
<td>get iterator to first element</td>
</tr>
<tr>
<td><code>Iterator End () const</code></td>
<td>get iterator past last element</td>
</tr>
<tr>
<td><code>Iterator Find (const TYPE &amp;val) const</code></td>
<td>find identical element in unsorted array (slow)</td>
</tr>
<tr>
<td><code>IndexT FindIndex (const TYPE &amp;val) const</code></td>
<td></td>
</tr>
</tbody>
</table>
**find index of identical element in unsorted array (slow)**

<table>
<thead>
<tr>
<th>void Sort ()</th>
<th>sort the array</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexT BinarySearchIndex (const TYPE &amp;val) const</td>
<td>do a binary search, requires a sorted array</td>
</tr>
</tbody>
</table>
template<class TYPE>
IndexT
Util::FixedArray< const
TYPE
>::BinarySearchIndex
const
(&

const TYPE elm )

Todo:

hmm, this is copy-pasted from Array...
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

**Util::FixedTable**
Util::FixedTable< TYPE > Class Template Reference

#include <fixedtable.h>
Detailed Description

template<class TYPE>
class Util::FixedTable< TYPE >

A fixed-size 2-dimensional array.

(C) 2006 Radon Labs GmbH
Public Member Functions

**FixedTable ()**
*default constructor*

**FixedTable (SizeT w, SizeT h)**
*constructor with size*

**FixedTable (SizeT w, SizeT h, const TYPE &val)**
*constructor with size and initialized contents*

**FixedTable (const FixedTable< TYPE > &rhs)**
*copy constructor*

**~FixedTable ()**
*destructor*

**void operator= (const FixedTable< TYPE > &rhs)**
*assignment operator*

**bool operator== (const FixedTable< TYPE > &rhs) const**
*equality operator*

**bool operator!= (const FixedTable< TYPE > &rhs) const**
*inequality operator*

**void SetSize (SizeT w, SizeT h)**
*set width and height (clears existing content)*

**SizeT Width () const**
*get width*

**SizeT Height () const**
*get height*

**void Clear (const TYPE &val)**
*clear the table with value*

**void Set (IndexT x, IndexT y, const TYPE &val)**
*set value at [x,y] position*

**TYPE & At (IndexT x, IndexT y) const**
*access value at [x,y] position*
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><code>FourCC()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>FourCC(uint f)</code></td>
<td>construct from 32-bit-value (e.g. <code>FourCC('ABCD')</code>)</td>
</tr>
<tr>
<td><code>FourCC(const String &amp;s)</code></td>
<td>construct from string</td>
</tr>
<tr>
<td><code>bool operator== (const FourCC &amp;rhs)</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>bool operator!= (const FourCC &amp;rhs)</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>bool operator&lt; (const FourCC &amp;rhs)</code></td>
<td>less-than operator</td>
</tr>
<tr>
<td><code>bool operator&lt;= (const FourCC &amp;rhs)</code></td>
<td>less-or-equal operator</td>
</tr>
<tr>
<td><code>bool operator&gt; (const FourCC &amp;rhs)</code></td>
<td>greater-than operator</td>
</tr>
<tr>
<td><code>bool operator&gt;= (const FourCC &amp;rhs)</code></td>
<td>greater-or-equal operator</td>
</tr>
<tr>
<td><code>bool IsValid()</code></td>
<td>return true if valid</td>
</tr>
<tr>
<td><code>void SetFromUInt(uint f)</code></td>
<td>set from 32-bit-value</td>
</tr>
<tr>
<td><code>uint AsUInt()</code></td>
<td>get as 32-bit-value</td>
</tr>
<tr>
<td><code>void SetFromString(const String &amp;s)</code></td>
<td>set as string</td>
</tr>
<tr>
<td><code>String AsString()</code></td>
<td>get as string</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <strong>String</strong> <strong>ToString</strong> (const <strong>FourCC</strong> &amp;f)</td>
<td>convert fourcc to string</td>
</tr>
<tr>
<td>static <strong>FourCC</strong> <strong>FromString</strong> (const <strong>String</strong> &amp;s)</td>
<td>convert string to fourcc</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:53 2008
Util::Guid
Util::Guid Class Reference

#include <guid.h>
Detailed Description

Implements a GUID.

(C) 2006 Radon Labs GmbH

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:53 2008
Util::HashTable
Util::HashTable< KEYTYPE, VALUETYPE > Class Template Reference

#include <hashtable.h>
Detailed Description

\[\text{template<class KEYTYPE, class VALUETYPE>}\]
\[\text{class Util::HashTable< KEYTYPE, VALUETYPE >}\]

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 then 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</strong> (SizeT capacity)</td>
<td>constructor with capacity</td>
</tr>
<tr>
<td><strong>HashTable</strong> (const HashTable&lt; KEYTYPE, VALUETYPE &gt; &amp;rhs)</td>
<td>copy constructor</td>
</tr>
<tr>
<td>void <strong>operator=</strong> (const HashTable&lt; KEYTYPE, VALUETYPE &gt; &amp;rhs)</td>
<td>assignment operator</td>
</tr>
<tr>
<td>VALUETYPE &amp; <strong>operator[]</strong> (const KEYTYPE &amp;key) const</td>
<td>read/write [] operator, assertion if key not found</td>
</tr>
<tr>
<td>SizeT <strong>Size</strong> () const</td>
<td>return current number of values in the hashtable</td>
</tr>
<tr>
<td>SizeT <strong>Capacity</strong> () const</td>
<td>return fixed-size capacity of the hash table</td>
</tr>
<tr>
<td>void <strong>Clear</strong> ()</td>
<td>clear the hashtable</td>
</tr>
<tr>
<td>bool <strong>IsEmpty</strong> () const</td>
<td>return true if empty</td>
</tr>
<tr>
<td>void <strong>Add</strong> (const KeyValuePair&lt; KEYTYPE, VALUETYPE &gt; &amp;kvp)</td>
<td>add a key/value pair object to the hash table</td>
</tr>
<tr>
<td>void <strong>Add</strong> (const KEYTYPE &amp;key, const VALUETYPE &amp;value)</td>
<td>add a key and associated value</td>
</tr>
<tr>
<td>void <strong>Erase</strong> (const KEYTYPE &amp;key)</td>
<td>erase an entry</td>
</tr>
<tr>
<td>bool <strong>Contains</strong> (const KEYTYPE &amp;key) const</td>
<td>return true if key exists in the array</td>
</tr>
<tr>
<td>Array&lt; KeyValuePair&lt; KEYTYPE, VALUETYPE &gt; &gt; <strong>Content</strong> () const</td>
<td>return array of all key/value pairs in the table (slow)</td>
</tr>
</tbody>
</table>
Util::KeyValuePair
#include <keyvaluepair.h>

Inheritance diagram for Util::KeyValuePair< KEYTYPE, VALUETYPE >:

![Inheritance Diagram](image-url)
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

- **KeyValuePair ()**  
  *default constructor*

- **KeyValuePair (const KEYTYPE &k, const VALUETYPE &v)**  
  *constructor with key and value*

- **KeyValuePair (const KEYTYPE &k)**  
  *constructor with key and undefined value*

- **KeyValuePair (const KeyValuePair<KEYTYPE, VALUETYPE> &rhs)**  
  *copy constructor*

- **void operator= (const KeyValuePair<KEYTYPE, VALUETYPE> &rhs)**  
  *assignment operator*

- **bool operator== (const KeyValuePair<KEYTYPE, VALUETYPE> &rhs) const**  
  *equality operator*

- **bool operator!= (const KeyValuePair<KEYTYPE, VALUETYPE> &rhs) const**  
  *inequality operator*

- **bool operator> (const KeyValuePair<KEYTYPE, VALUETYPE> &rhs) const**  
  *greater operator*

- **bool operator>= (const KeyValuePair<KEYTYPE, VALUETYPE> &rhs) const**  
  *greater-or-equal operator*

- **bool operator< (const KeyValuePair<KEYTYPE, VALUETYPE> &rhs) const**  
  *lesser operator*

- **bool operator<= (const KeyValuePair<KEYTYPE, VALUETYPE> &rhs) const**  
  *lesser-or-equal operator*

- **VALUETYPE & Value ()**  
  *read/write access to value*
<table>
<thead>
<tr>
<th>const KEYTYPE &amp;</th>
<th>Key () const</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>read access to key</td>
</tr>
<tr>
<td>const VALUETYPE &amp;</td>
<td>Value () const</td>
</tr>
<tr>
<td></td>
<td>read access to key</td>
</tr>
</tbody>
</table>
Constructor & Destructor Documentation

template<class KEYTYPE, class VALUETYPE>
Util::KeyValuePair<
  const
  (KEYTYPE k ) [inline]
&

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::LightFlickerUtil
Util::LightFlickerUtil Class Reference

#include <lightflickerutil.h>
Detailed Description

Manipulate a light entity with position and intensity flickering.

(C) 2007 Radon Labs GmbH
Util::List
Util::List< TYPE > Class Template Reference

#include <list.h>

Inheritance diagram for Util::List< TYPE >:
Detailed Description

template<class TYPE>
class Util::List< TYPE >

Implements a doubly linked list.

(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><strong>Iterator</strong></td>
<td><strong>Begin</strong> () const</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------</td>
</tr>
<tr>
<td><strong>Iterator</strong></td>
<td><strong>End</strong> () const</td>
</tr>
<tr>
<td><strong>Iterator</strong></td>
<td><strong>Find</strong> (const TYPE &amp;e, <strong>Iterator</strong> start) const</td>
</tr>
<tr>
<td></td>
<td></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><strong>Iterator ()</strong></td>
<td>default constructor</td>
</tr>
<tr>
<td>*<em>Iterator (Node <em>node)</em></em></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>Iterator (const Iterator &amp;rhs)</strong></td>
<td>copy constructor</td>
</tr>
<tr>
<td><strong>const Iterator &amp; operator= (const Iterator &amp;rhs)</strong></td>
<td>assignment operator</td>
</tr>
<tr>
<td><strong>bool operator== (const Iterator &amp;rhs)</strong></td>
<td>const equality operator</td>
</tr>
<tr>
<td><strong>bool operator!= (const Iterator &amp;rhs)</strong></td>
<td>const inequality operator</td>
</tr>
<tr>
<td><strong>const Iterator &amp; operator++ ()</strong></td>
<td>pre-increment operator</td>
</tr>
<tr>
<td><strong>Iterator operator++ (int)</strong></td>
<td>post-increment operator</td>
</tr>
<tr>
<td><strong>const Iterator &amp; operator-- ()</strong></td>
<td>pre-decrement operator</td>
</tr>
<tr>
<td><strong>Iterator operator-- (int)</strong></td>
<td>post-increment operator</td>
</tr>
<tr>
<td><strong>operator bool ()</strong></td>
<td>const bool operator</td>
</tr>
<tr>
<td><strong>TYPE * operator-&gt; ()</strong></td>
<td>const safe -&gt; operator</td>
</tr>
<tr>
<td><strong>TYPE &amp; operator * ()</strong></td>
<td>const safe dereference operator</td>
</tr>
</tbody>
</table>
Util::Proxy
Util::Proxy< TYPE > Class Template Reference

#include <proxy.h>
Detailed Description

template<class TYPE>
class Util::Proxy< TYPE >

A Proxy holds an embedded object of any class and offers compare operators which are routed to the target object. Several Proxies may point to the same target object, a refcount is associated with the target object, and it will automatically be destroyed when all Proxies have gone. Proxies are useful in some situations when working with container classes, since they guarantee that the target object doesn't move in memory even when the proxy itself is moved around by the container class.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proxy ()</strong></td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><strong>Proxy (const TYPE &amp;rhs)</strong></td>
<td><em>construct from type</em></td>
</tr>
<tr>
<td><strong>Proxy (const Proxy&lt; TYPE &gt; &amp;rhs)</strong></td>
<td><em>copy constructor</em></td>
</tr>
<tr>
<td>void operator=(const TYPE &amp;rhs)</td>
<td><em>assignment operator from type</em></td>
</tr>
<tr>
<td>void operator=(const Proxy&lt; TYPE &gt; &amp;rhs)</td>
<td><em>assignment operator from proxy</em></td>
</tr>
<tr>
<td>bool operator==(const TYPE &amp;rhs)</td>
<td><em>equality operator with type</em></td>
</tr>
<tr>
<td>bool operator!=(const TYPE &amp;rhs)</td>
<td><em>inequality operator with type</em></td>
</tr>
<tr>
<td>bool operator&gt;(const TYPE &amp;rhs)</td>
<td><em>greater-than operator with type</em></td>
</tr>
<tr>
<td>bool operator&lt;(const TYPE &amp;rhs)</td>
<td><em>less-than operator with type</em></td>
</tr>
<tr>
<td>bool operator&gt;=(const TYPE &amp;rhs)</td>
<td><em>greater-or-equal operator with type</em></td>
</tr>
<tr>
<td>bool operator&lt;=(const TYPE &amp;rhs)</td>
<td><em>less-or-equal operator with type</em></td>
</tr>
<tr>
<td>bool operator==(const Proxy&lt; TYPE &gt; &amp;rhs)</td>
<td><em>equality operator with proxy</em></td>
</tr>
<tr>
<td>bool operator!=(const Proxy&lt; TYPE &gt; &amp;rhs)</td>
<td><em>inequality operator with proxy</em></td>
</tr>
<tr>
<td>bool operator&gt;(const Proxy&lt; TYPE &gt; &amp;rhs)</td>
<td><em>greater-than operator with proxy</em></td>
</tr>
<tr>
<td>bool operator&lt;(const Proxy&lt; TYPE &gt; &amp;rhs)</td>
<td><em>less-than operator with proxy</em></td>
</tr>
<tr>
<td>bool operator&gt;=(const Proxy&lt; TYPE &gt; &amp;rhs)</td>
<td><em>greater-or-equal operator with proxy</em></td>
</tr>
<tr>
<td>bool operator&lt;=(const Proxy&lt; TYPE &gt; &amp;rhs)</td>
<td><em>less-or-equal operator with proxy</em></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>const TYPE &amp; GetObject () const</code></td>
<td>Access to contained object</td>
</tr>
<tr>
<td><code>int GetObjectRefCount () const</code></td>
<td>Get refcount of object</td>
</tr>
<tr>
<td><code>bool IsValid () const</code></td>
<td>Return true if the proxy contains a valid object</td>
</tr>
<tr>
<td><code>void Clear ()</code></td>
<td>Clear content</td>
</tr>
</tbody>
</table>
Constructor & Destructor Documentation

template<class TYPE>

Util::Proxy< const
TYPE ( TYPE rhs ) [inline, explicit]
> :: Proxy &

construct from type

Constructing a proxy from TYPE always creates a new target object.

template<class TYPE>

Util::Proxy< const
TYPE ( Proxy< TYPE rhs > ) [inline, explicit]
> :: Proxy > &

copy constructor

Constructing a proxy from another proxy shares the target object of the right-hand-side proxy and increments its refcount.
Member Function Documentation

template<class TYPE>
void
Util::Proxy< TYPE > const
  ( TYPE rhs ) [inline]
>::operator= &

assignment operator from type

Assigning a TYPE to the proxy will throw away the previous target object and create a new one.

template<class TYPE>
void
const
Util::Proxy< Proxy< TYPE > rhs > [inline]
>::operator= > &

assignment operator from proxy

Assigning a proxy to this proxy will share the right-hand-side's target object and increment its refcount.
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
<table>
<thead>
<tr>
<th>Public Member Functions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node ()</strong></td>
<td></td>
</tr>
<tr>
<td><code>constructor</code></td>
<td></td>
</tr>
<tr>
<td><strong>~Node ()</strong></td>
<td></td>
</tr>
<tr>
<td><code>destructor</code></td>
<td></td>
</tr>
<tr>
<td>*<em>void Initialize (QuadTree&lt; TYPE &gt; <em>tree, uchar _level, ushort _col, ushort _row)</em></em></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 ClearDataPtr ()</strong></td>
<td>clear element ptr</td>
</tr>
<tr>
<td>*<em>void SetDataPtr (TYPE <em>elem)</em></em></td>
<td>set element ptr</td>
</tr>
</tbody>
</table>
Member Function Documentation

template<class TYPE>
void
Util::QuadTree< TYPE >::Node::Initialize ( QuadTree< TYPE >* tree,
uchar _level,
ushort _col,
ushort _row
) [inline]

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 ) [inline]

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 >:

```
Util::Queue< TYPE >
```

```
Threading::SafeQueue< TYPE >
```

```
< TYPE >._map" border="0" alt=""
< TYPE >._map"> < TYPE >" shape="rect" coords="0,56,195,80"
```
Detailed Description

template<class TYPE>
class Util::Queue< TYPE >

Nebula3’s queue class (a FIFO container).

(C) 2006 Radon Labs GmbH
Public Member Functions

```
Queue ()
constructor

Queue (const Queue< TYPE > &rhs)
copy constructor

void operator= (const Queue< TYPE > &rhs)
assignment operator

TYPE & operator[] (IndexT index) const
access element by index, 0 is the frontmost element (next to be dequeued)

bool operator== (const Queue< TYPE > &rhs) const
equality operator

bool operator!= (const Queue< TYPE > &rhs) const
inequality operator

SizeT Size () const
returns number of elements in the queue

bool IsEmpty () const
return true if queue is empty

void Clear ()
remove all elements from the queue

bool Contains (const TYPE &e) const
return true if queue contains element

void Enqueue (const TYPE &e)
add element to the back of the queue

TYPE Dequeue ()
remove the element from the front of the queue

TYPE & Peek () const
access to element at front of queue without removing it
```
Util::SegmentedGfxUtil
Util::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
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><strong>SimpleTree</strong> ()</td>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><strong>Node &amp; Root</strong> ()</td>
<td><em>read/write access to root element</em></td>
</tr>
<tr>
<td>const <strong>Node &amp; Root</strong> () const</td>
<td><em>read-only access to root element</em></td>
</tr>
</tbody>
</table>
Data Structures

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
<table>
<thead>
<tr>
<th>Public Member Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node ()</strong></td>
</tr>
<tr>
<td><em>default constructor</em></td>
</tr>
<tr>
<td><strong>Node (const Node &amp;parent, const VALUETYPE &amp;val)</strong></td>
</tr>
<tr>
<td><em>constructor with parent and value</em></td>
</tr>
<tr>
<td><strong>~Node ()</strong></td>
</tr>
<tr>
<td><em>destructor</em></td>
</tr>
<tr>
<td>void <strong>AddRef ()</strong></td>
</tr>
<tr>
<td><em>increment refcount</em></td>
</tr>
<tr>
<td>void <strong>Release ()</strong></td>
</tr>
<tr>
<td><em>decrement refcount</em></td>
</tr>
<tr>
<td>const Node &amp; <strong>operator[](IndexT i) const</strong></td>
</tr>
<tr>
<td><em>get read-only child by index</em></td>
</tr>
<tr>
<td>Node &amp; <strong>operator[](IndexT i)</strong></td>
</tr>
<tr>
<td><em>get read/write child by index</em></td>
</tr>
<tr>
<td>const Node &amp; <strong>Child (IndexT i) const</strong></td>
</tr>
<tr>
<td><em>get read-only child element at index</em></td>
</tr>
<tr>
<td>Node &amp; <strong>Child (IndexT i)</strong></td>
</tr>
<tr>
<td><em>get read/write child element at index</em></td>
</tr>
<tr>
<td>bool <strong>HasParent () const</strong></td>
</tr>
<tr>
<td><em>return true if the node has a parent</em></td>
</tr>
<tr>
<td>Node &amp; <strong>Parent ()</strong></td>
</tr>
<tr>
<td><em>read/write access to parent</em></td>
</tr>
<tr>
<td>const Node &amp; <strong>Parent () const</strong></td>
</tr>
<tr>
<td><em>read-only access to parent</em></td>
</tr>
<tr>
<td>void <strong>Clear ()</strong></td>
</tr>
<tr>
<td><em>clear children</em></td>
</tr>
<tr>
<td>SizeT <strong>Size () const</strong></td>
</tr>
<tr>
<td><em>number of children</em></td>
</tr>
<tr>
<td>bool <strong>IsEmpty () const</strong></td>
</tr>
<tr>
<td><em>return true if empty</em></td>
</tr>
<tr>
<td>Node &amp; <strong>Front () const</strong></td>
</tr>
<tr>
<td><em>return first element</em></td>
</tr>
<tr>
<td>Node &amp; Back () const</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>void Append (const VALUETYPE &amp;val)</td>
</tr>
<tr>
<td>void Erase (IndexT i)</td>
</tr>
<tr>
<td>void Insert (IndexT index, const VALUETYPE &amp;val)</td>
</tr>
<tr>
<td>IndexT Find (const VALUETYPE &amp;val) const</td>
</tr>
<tr>
<td>VALUETYPE &amp; Value ()</td>
</tr>
<tr>
<td>const VALUETYPE &amp; Value () const</td>
</tr>
</tbody>
</table>
Util::Stack
Util::Stack< TYPE > Class Template Reference

#include <stack.h>
Detailed Description

```cpp
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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stack ()</strong></td>
<td><em>constructor</em></td>
</tr>
<tr>
<td><strong>Stack (const Stack&lt; TYPE &gt; &amp;rhs)</strong></td>
<td><em>copy constructor</em></td>
</tr>
<tr>
<td><strong>operator= (const Stack&lt; TYPE &gt; &amp;rhs)</strong></td>
<td><em>assignment operator</em></td>
</tr>
<tr>
<td><strong>TYPE &amp;  operator[] (IndexT index) const</strong></td>
<td><em>access element by index, 0 is the topmost element</em></td>
</tr>
<tr>
<td><strong>bool operator== (const Stack&lt; TYPE &gt; &amp;rhs) const</strong></td>
<td><em>equality operator</em></td>
</tr>
<tr>
<td><strong>bool operator!= (const Stack&lt; TYPE &gt; &amp;rhs) const</strong></td>
<td><em>inequality operator</em></td>
</tr>
<tr>
<td><strong>SizeT Size () const</strong></td>
<td><em>returns number of elements on stack</em></td>
</tr>
<tr>
<td><strong>bool isEmpty () const</strong></td>
<td><em>returns true if stack is empty</em></td>
</tr>
<tr>
<td><strong>void Clear ()</strong></td>
<td><em>remove all elements from the stack</em></td>
</tr>
<tr>
<td><strong>bool Contains (const TYPE &amp;e) const</strong></td>
<td><em>return true if stack contains element</em></td>
</tr>
<tr>
<td><strong>void Push (const TYPE &amp;e)</strong></td>
<td><em>push an element on the stack</em></td>
</tr>
<tr>
<td><strong>TYPE &amp;  Peek () const</strong></td>
<td><em>get reference of topmost element of stack, without removing it</em></td>
</tr>
<tr>
<td><strong>TYPE Pop ()</strong></td>
<td><em>get topmost element of stack, remove element</em></td>
</tr>
</tbody>
</table>
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><code>String()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>String (const String &amp;rhs)</code></td>
<td>Copy constructor</td>
</tr>
<tr>
<td><code>String (const char *cStr)</code></td>
<td>Construct from C string</td>
</tr>
<tr>
<td><code>~String()</code></td>
<td>Destructor</td>
</tr>
<tr>
<td><code>void * operator new (size_t size)</code></td>
<td>Overloaded operator new</td>
</tr>
<tr>
<td><code>void operator delete (void *)</code></td>
<td>Overloaded operator delete</td>
</tr>
<tr>
<td><code>void operator= (const String &amp;rhs)</code></td>
<td>Assignment operator</td>
</tr>
<tr>
<td><code>void operator= (const char *cStr)</code></td>
<td>Assign from const char*</td>
</tr>
<tr>
<td><code>void operator+= (const String &amp;rhs)</code></td>
<td>+= operator</td>
</tr>
<tr>
<td><code>const char operator[] (IndexT i) const</code></td>
<td>Read-only index operator</td>
</tr>
<tr>
<td><code>char &amp; operator[] (IndexT i)</code></td>
<td>Read/write index operator</td>
</tr>
<tr>
<td><code>void Reserve (SizeT newSize)</code></td>
<td>Reserve internal buffer size to prevent heap allocs</td>
</tr>
<tr>
<td><code>SizeT Length () const</code></td>
<td>Return length of string</td>
</tr>
<tr>
<td><code>void Clear ()</code></td>
<td>Clear the string</td>
</tr>
<tr>
<td><code>bool IsEmpty () const</code></td>
<td>Return true if string object is empty</td>
</tr>
<tr>
<td><code>bool IsValid () const</code></td>
<td>Return true if string object is not empty</td>
</tr>
<tr>
<td><code>void Append (const String &amp;str)</code></td>
<td>Append the string</td>
</tr>
</tbody>
</table>
append string

void Append (const char *str)
append c-string

void AppendRange (const char *str, SizeT numChars)
append a range of characters

void ToLower ()
convert string to lower case

void ToUpper ()
convert string to upper case

Array<String> Tokenize (const String &whiteSpace) const
tokenize string into a provided String array

Array<String> Tokenize (const String &whiteSpace, char fence) const
tokenize string, keep strings within fence characters intact

String ExtractRange (IndexT fromIndex, SizeT numChars) const
extract substring

String ExtractToEnd (IndexT fromIndex) const
extract substring to end of this string

void Strip (const String &charSet)
terminate string at first occurrence of character in set

IndexT FindStringIndex (const String &s, IndexT startIndex=0) const
return start index of substring, or InvalidIndex if not found

IndexT FindCharIndex (char c, IndexT startIndex=0) const
return index of character in string, or InvalidIndex if not found

void TerminateAtIndex (IndexT index)
terminate string at given index

bool ContainsCharFromSet (const String &charSet) const
returns true if string contains any character from set

void TrimLeft (const String &charSet)
delete characters from charset at left side of string

void TrimRight (const String &charSet)
delete characters from charset at right side of string

void Trim (const String &charSet)
trim characters from charset at both sides of string

void SubstituteString (const String &str, const String
<table>
<thead>
<tr>
<th>void</th>
<th>&amp;substStr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>substitute every occurrence of a string with another string</td>
</tr>
<tr>
<td>void</td>
<td>SubstituteChar (char c, char subst)</td>
</tr>
<tr>
<td></td>
<td>substitute every occurrence of a character with another character</td>
</tr>
<tr>
<td>void __cdecl Format</td>
<td>(const char *fmtString,....)</td>
</tr>
<tr>
<td></td>
<td>format string printf-style</td>
</tr>
<tr>
<td>void __cdecl FormatArgList</td>
<td>(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>CheckValidCharSet (const String &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>ReplaceChars (const String &amp;charSet, char replacement)</td>
</tr>
<tr>
<td></td>
<td>replace any character set character within a string with the replacement character</td>
</tr>
<tr>
<td>IndexT</td>
<td>HashCode () const</td>
</tr>
<tr>
<td></td>
<td>return a 32-bit hash code for the string</td>
</tr>
<tr>
<td>void</td>
<td>UTF8toANSI ()</td>
</tr>
<tr>
<td></td>
<td>convert string inplace from UTF-8 to 8-bit ANSI</td>
</tr>
<tr>
<td>void</td>
<td>ANSItoUTF8 ()</td>
</tr>
<tr>
<td></td>
<td>convert ANSI to UTF-8 in place</td>
</tr>
<tr>
<td>void</td>
<td>SetCharPtr (const char *s)</td>
</tr>
<tr>
<td></td>
<td>set content to char ptr</td>
</tr>
<tr>
<td>void</td>
<td>Set (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>SetInt (int val)</td>
</tr>
<tr>
<td></td>
<td>set as int value</td>
</tr>
<tr>
<td>void</td>
<td>SetFloat (float val)</td>
</tr>
<tr>
<td></td>
<td>set as float value</td>
</tr>
<tr>
<td>void</td>
<td>SetBool (bool val)</td>
</tr>
<tr>
<td></td>
<td>set as bool value</td>
</tr>
<tr>
<td>void</td>
<td>SetFloat4 (const Math::float4 &amp;v)</td>
</tr>
<tr>
<td></td>
<td>set as float4 value</td>
</tr>
<tr>
<td>void</td>
<td>SetMatrix44 (const Math::matrix44 &amp;v)</td>
</tr>
<tr>
<td></td>
<td>set as matrix44 value</td>
</tr>
<tr>
<td>void</td>
<td>AppendInt (int val)</td>
</tr>
<tr>
<td></td>
<td>append int value</td>
</tr>
<tr>
<td>Function Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>void</td>
<td><strong>AppendFloat</strong> (float val) append float value</td>
</tr>
<tr>
<td>void</td>
<td><strong>AppendBool</strong> (bool val) append bool value</td>
</tr>
<tr>
<td>void</td>
<td><strong>AppendFloat4</strong> (const Math::float4 &amp;v) append float4 value</td>
</tr>
<tr>
<td>void</td>
<td><strong>AppendMatrix44</strong> (const Math::matrix44 &amp;v) append matrix44 value</td>
</tr>
<tr>
<td>const char *</td>
<td><strong>AsCharPtr</strong> () const return contents as character pointer</td>
</tr>
<tr>
<td>int</td>
<td><strong>AsInt</strong> () const return contents as integer</td>
</tr>
<tr>
<td>float</td>
<td><strong>AsFloat</strong> () const return contents as float</td>
</tr>
<tr>
<td>bool</td>
<td><strong>AsBool</strong> () const return contents as bool</td>
</tr>
<tr>
<td>Math::float4</td>
<td><strong>AsFloat4</strong> () const return contents as float4</td>
</tr>
<tr>
<td>Math::matrix44</td>
<td><strong>AsMatrix44</strong> () const return contents as matrix44</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsValidInt</strong> () const return true if the content is a valid integer</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsValidFloat</strong> () const return true if the content is a valid float</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsValidBool</strong> () const return true if the content is a valid bool</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsValidFloat4</strong> () const return true if the content is a valid float4</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsValidMatrix44</strong> () const return true if content is a valid matrix44</td>
</tr>
<tr>
<td>String</td>
<td><strong>GetFileExtension</strong> () const get filename extension without dot</td>
</tr>
<tr>
<td>bool</td>
<td><strong>CheckFileExtension</strong> (const String &amp;ext) const check file extension</td>
</tr>
<tr>
<td>void</td>
<td><strong>ConvertBackslashes</strong> () convert backslashes to slashes</td>
</tr>
<tr>
<td>Type</td>
<td>Function Name</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>void</td>
<td>StripFileExtension()</td>
</tr>
<tr>
<td>String</td>
<td>ExtractFileName() const</td>
</tr>
<tr>
<td>String</td>
<td>ExtractLastDirName() const</td>
</tr>
<tr>
<td>String</td>
<td>ExtractDirName() const</td>
</tr>
<tr>
<td>String</td>
<td>ExtractToLastSlash() const</td>
</tr>
<tr>
<td>void</td>
<td>ReplaceIllegalFilenameChars(char replacement)</td>
</tr>
</tbody>
</table>
**Static 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>static void</td>
<td>Setup</td>
<td>()</td>
<td>Static Setup method called by Util::Setup</td>
</tr>
<tr>
<td>static String</td>
<td>Concatenate</td>
<td>(const Array&lt; String &gt; &amp;strArray, const String &amp;whiteSpace)</td>
<td>Concatenate array of strings into new string</td>
</tr>
<tr>
<td>static bool</td>
<td>MatchPattern</td>
<td>(const String &amp;str, const String &amp;pattern)</td>
<td>Pattern matching</td>
</tr>
<tr>
<td>static String</td>
<td>FromInt</td>
<td>(int i)</td>
<td>Construct a string from an int</td>
</tr>
<tr>
<td>static String</td>
<td>FromFloat</td>
<td>(float f)</td>
<td>Construct a string from a float</td>
</tr>
<tr>
<td>static String</td>
<td>FromBool</td>
<td>(bool b)</td>
<td>Construct a string from a bool</td>
</tr>
<tr>
<td>static String</td>
<td>FromFloat4</td>
<td>(const Math::float4 &amp;v)</td>
<td>Construct a string from float4</td>
</tr>
<tr>
<td>static String</td>
<td>FromMatrix44</td>
<td>(const Math::matrix44 &amp;m)</td>
<td>Construct a string from matrix44</td>
</tr>
</tbody>
</table>
**Friends**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>operator== (const String &amp;a, const String &amp;b)</code></td>
<td>equality operator</td>
</tr>
<tr>
<td><code>operator== (const String &amp;a, const char *cStr)</code></td>
<td>shortcut equality operator</td>
</tr>
<tr>
<td><code>operator== (const char *cStr, const String &amp;a)</code></td>
<td>shortcut equality operator</td>
</tr>
<tr>
<td><code>operator!= (const String &amp;a, const String &amp;b)</code></td>
<td>inequality operator</td>
</tr>
<tr>
<td><code>operator&lt; (const String &amp;a, const String &amp;b)</code></td>
<td>less-than operator</td>
</tr>
<tr>
<td><code>operator&gt; (const String &amp;a, const String &amp;b)</code></td>
<td>greater-than operator</td>
</tr>
<tr>
<td><code>operator&lt;= (const String &amp;a, const String &amp;b)</code></td>
<td>less-or-equal operator</td>
</tr>
<tr>
<td><code>operator&gt;= (const String &amp;a, const String &amp;b)</code></td>
<td>greater-then operator</td>
</tr>
</tbody>
</table>
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.

Array<String>
Util::String::Tokenize(const String & whiteSpace) const

tokenize string into a provided String array

Tokenize the string into a String array.

Parameters:

whiteSpace a string containing the whitespace characters

Returns:

a string array of tokens

Array<String>
Util::String::Tokenize(const String whiteSpace, & char fence) const

tokenize string, keep strings within fence characters intact

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.

String
Util::String::ExtractRange ( IndexT from,
                    SizeT numChars
            ) const

extract substring

Extract a substring range.

String
Util::String::ExtractToEnd ( IndexT fromIndex ) const

extract substring to end of this string

Extract 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.

void
Util::String::TerminateAtIndex (IndexT index) [inline]

terminate string at given index

Terminates the string at the given index.

bool
Util::String::ContainsCharFromSet (const String& charSet) const [inline]

returns true if string contains any character from set

Returns true if string contains one of the characters from charset.

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.

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.

void
Util::String::Trim (const String& charSet)
trim characters from charset at both sides of string

Trim both sides of a string.

```c
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.

```c
void Util::String::SubstituteChar (char c, char subst) [inline]
```

substitute every occurrence of a character with another character

Replace character with another.

```c
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.

```c
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.

void
Util::String::UTF8toANSI ( )

convert string inplace from UTF-8 to 8-bit ANSI

This converts an UTF-8 string to 8-bit-ANSI. Note that only characters in the range 0 .. 255 are converted, all other characters will be converted to a question mark.

For conversion rules see http://www.cl.cam.ac.uk/~mgk25/unicode.html#utf-8

void
Util::String::ANSItoUTF8 ( )

convert ANSI to UTF-8 in place

Convert contained ANSI string to UTF-8 in place.

void
Util::String::Set ( const 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.

int
Util::String::AsInt ( ) const [inline]
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!

```cpp
float
Util::String::AsFloat() const [inline]
```

return contents as float

Returns content as float. Note: this method doesn't check whether the contents is actually a valid float. Use the IsValidInt() method for this!

```cpp
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!

```cpp
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!

```cpp
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.

```cpp
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.

```cpp
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 [inline]
```

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() [inline]
```

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.
**`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

**`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

**`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 shortcut typedef for Atom<String>.

(C) 2007 Radon Labs GmbH
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Util::Variant
#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

```markdown
type

<table>
<thead>
<tr>
<th>enum</th>
<th><strong>Type</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>variant types</td>
</tr>
</tbody>
</table>
```
## Public Member Functions

<table>
<thead>
<tr>
<th>Variant</th>
<th>()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>default constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const Variant &amp;rhs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>copy constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(int rhs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>int constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(float rhs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>float constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(bool rhs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bool constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const Math::float4 &amp;v)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>float4 constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const Math::matrix44 &amp;m)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>matrix44 constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const Util::String &amp;rhs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>string constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const Util::Blob &amp;blob)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>blob constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const Util::Guid &amp;guid)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>guid constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const char *chrPtr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>const char constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(Core::RefCounted *ptr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>object constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const Util::Array&lt; int &gt; &amp;rhs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>int array constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const Util::Array&lt; float &gt; &amp;rhs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>float array constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const Util::Array&lt; bool &gt; &amp;rhs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bool array constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const Util::Array&lt; Math::float4 &gt; &amp;rhs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>float4 array constructor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variant</th>
<th>(const Util::Array&lt; Math::matrix44 &gt;</th>
</tr>
</thead>
</table>
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 ()
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)
string assignment operator

void operator= (const Util::Blob &val)
blob assignment operator

void operator= (const Util::Guid &val)
guid assignment operator

void operator= (const char *chrPtr)
char pointer assignment
void operator= (Core::RefCounted *ptr)
object assignment

void operator= (const Util::Array<int> &rhs)
int array assignment

void operator= (const Util::Array<float> &rhs)
float array assignment

void operator= (const Util::Array<bool> &rhs)
bool array assignment

void operator= (const Util::Array<Math::float4> &rhs)
float4 array assignment

void operator= (const Util::Array<Math::matrix44> &rhs)
matrix44 array assignment

void operator= (const Util::Array<Util::String> &rhs)
string array assignment

void operator= (const Util::Array<Util::Blob> &rhs)
blob array assignment

void operator= (const Util::Array<Util::Guid> &rhs)
guid array assignment

bool operator== (const Variant &rhs) const
equality operator

bool operator== (int rhs) const
int equality operator

bool operator== (float rhs) const
float equality operator

bool operator== (bool rhs) const
bool equality operator

bool operator== (const Math::float4 &rhs) const
float4 equality operator

bool operator== (const Util::String &rhs) const
string equality operator

bool operator== (const Util::Guid &rhs) const
guid equality operator

bool operator== (const char *chrPtr) const
char ptr equality operator
bool operator== (Core::RefCounted *ptr) const
pointer equality operator

bool operator!=(const Variant &rhs) const
inequality operator

bool operator!=(int rhs) const
int inequality operator

bool operator!=(float rhs) const
float inequality operator

bool operator!=(bool rhs) const
bool inequality operator

bool operator!=(const Math::float4 &rhs) const
float4 inequality operator

bool operator!=(const Util::String &rhs) const
string inequality operator

bool operator!=(const Util::Guid &rhs) const
guid inequality operator

bool operator!=(const char *chrPtr) const
char ptr inequality operator

bool operator!=(Core::RefCounted *ptr) const
pointer equality 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
void SetFloat4 (const Math::float4 &val)
set float4 content

Math::float4 GetFloat4 () const
get float4 content

void SetMatrix44 (const Math::matrix44 &val)
set matrix44 content

const Math::matrix44 & GetMatrix44 () const
get matrix44 content

void SetBlob (const Util::Blob &val)
set blob

const Util::Blob & GetBlob () const
get blob

void SetGuid (const Util::Guid &val)
set guid content

const Util::Guid & GetGuid () const
get guid content

void SetObject (Core::RefCounted *ptr)
set object pointer

Core::RefCounted * GetObject () const
get object pointer

void SetIntArray (const Util::Array<int> &val)
set int array content

const Util::Array<int> & GetIntArray () const
get int array content

void SetFloatArray (const Util::Array<float> &val)
set float array content

const Util::Array<float> & GetFloatArray () const
get float array content

void SetBoolArray (const Util::Array<bool> &val)
set bool array content

const Util::Array<bool> & GetBoolArray () const
get bool array content

void SetFloat4Array (const Util::Array<Math::float4> &val)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set float4 array content</td>
<td>const Util::Array<a href="">Math::float4</a> &amp; GetFloat4Array() const</td>
</tr>
<tr>
<td>get float4 array content</td>
<td></td>
</tr>
<tr>
<td>SetMatrix44Array (const Util::Array<a href="">Math::matrix44</a> &amp;val)</td>
<td>void</td>
</tr>
<tr>
<td>set matrix44 array content</td>
<td></td>
</tr>
<tr>
<td>SetStringArray (const Util::Array<a href="">Util::String</a> &amp;val)</td>
<td>void</td>
</tr>
<tr>
<td>set string array content</td>
<td></td>
</tr>
<tr>
<td>SetGuidArray (const Util::Array<a href="">Util::Guid</a> &amp;val)</td>
<td>void</td>
</tr>
<tr>
<td>set guid array content</td>
<td></td>
</tr>
<tr>
<td>SetBlobArray (const Util::Array<a href="">Util::Blob</a> &amp;val)</td>
<td>void</td>
</tr>
<tr>
<td>set blob array content</td>
<td></td>
</tr>
<tr>
<td>GetBlobArray () const</td>
<td>const Util::Array<a href="">Util::Blob</a> &amp; GetBlobArray()</td>
</tr>
</tbody>
</table>
Static Public Member Functions

| static Util::String TypeToString (Type t) | convert type to string |
| static Type StringToType (const Util::String &str) | convert string to type |

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:53 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

**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>static void Setup()</code></td>
<td>Setup lowlevel static objects (must be called before spawning any threads)</td>
</tr>
<tr>
<td><code>static void Exit(int exitCode)</code></td>
<td>Exit process, and to proper cleanup, memleak reporting, etc...</td>
</tr>
<tr>
<td><code>static void Error(const char *error)</code></td>
<td>Display an error message box</td>
</tr>
<tr>
<td><code>static void DebugOut(const char *msg)</code></td>
<td>Print a message on the debug console</td>
</tr>
<tr>
<td><code>static void Sleep(double sec)</code></td>
<td>Sleep for a specified amount of seconds</td>
</tr>
<tr>
<td><code>static Util::String GetLastError()</code></td>
<td>Returns GetLastError() as string</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

Win32::Win32Barrier
Win32::Win32Barrier Class Reference

#include <win32barrier.h>
Detailed Description

Implements the 2 macros ReadWriteBarrier and MemoryBarrier.

ReadWriteBarrier prevents the compiler from re-ordering memory accesses accross 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
Win32::Win32CalendarTime
Win32::Win32CalendarTime Class Reference

#include <win32calendar_time.h>

Inheritance diagram for Win32::Win32CalendarTime:
Detailed Description

Win32 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><code>Month</code></td>
</tr>
<tr>
<td></td>
<td><code>months enum</code></td>
</tr>
<tr>
<td>enum</td>
<td><code>Weekday</code></td>
</tr>
<tr>
<td></td>
<td><code>weekdays enum</code></td>
</tr>
<tr>
<td>typedef</td>
<td><code>Year</code></td>
</tr>
<tr>
<td></td>
<td><code>typedefs</code></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td><code>void SetYear (Year y)</code></td>
<td>set the year</td>
</tr>
<tr>
<td><code>Year GetYear () const</code></td>
<td>get the year</td>
</tr>
<tr>
<td><code>void SetMonth (Month m)</code></td>
<td>set the month</td>
</tr>
<tr>
<td><code>Month GetMonth () const</code></td>
<td>get the month</td>
</tr>
<tr>
<td><code>void SetWeekday (Weekday wd)</code></td>
<td>set the day-of-week</td>
</tr>
<tr>
<td><code>Weekday GetWeekday () const</code></td>
<td>get the day-of-week</td>
</tr>
<tr>
<td><code>void SetDay (Day d)</code></td>
<td>set the day (of month)</td>
</tr>
<tr>
<td><code>Day GetDay () const</code></td>
<td>get the day (of month)</td>
</tr>
<tr>
<td><code>void SetHour (Hour h)</code></td>
<td>set hour-of-day</td>
</tr>
<tr>
<td><code>Hour GetHour () const</code></td>
<td>get hour-of-day</td>
</tr>
<tr>
<td><code>void SetMinute (Minute m)</code></td>
<td>set minute-of-hour</td>
</tr>
<tr>
<td><code>Minute GetMinute () const</code></td>
<td>get minute-of-hour</td>
</tr>
<tr>
<td><code>void SetSecond (Second s)</code></td>
<td>set second-of-minute</td>
</tr>
<tr>
<td><code>Second GetSecond () const</code></td>
<td>get second-of-minute</td>
</tr>
<tr>
<td><code>void SetMilliSecond (MilliSecond ms)</code></td>
<td>set milliseconds</td>
</tr>
<tr>
<td><code>MilliSecond GetMilliSecond () const</code></td>
<td>get milliseconds</td>
</tr>
</tbody>
</table>
### Static Public Member Functions

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

**CalendarTime**
Win32::Win32CalendarTime::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**
Win32::Win32CalendarTime::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
Win32::Win32ConsoleHandler
Win32::Win32ConsoleHandler Class Reference

#include <win32consolehandler.h>

Inheritance diagram for Win32::Win32ConsoleHandler:

```
Core::RefCounted

IO::ConsoleHandler

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><code>Win32ConsoleHandler()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual void <code>Print</code> (const <code>Util::String</code> &amp;s)</td>
<td>called by console to output data</td>
</tr>
<tr>
<td>virtual void <code>Error</code> (const <code>Util::String</code> &amp;s)</td>
<td>called by console with serious error</td>
</tr>
<tr>
<td>virtual void <code>Warning</code> (const <code>Util::String</code> &amp;s)</td>
<td>called by console to output warning</td>
</tr>
<tr>
<td>virtual void <code>DebugOut</code> (const <code>Util::String</code> &amp;s)</td>
<td>called by console to output debug string</td>
</tr>
<tr>
<td>virtual bool <code>HasInput</code> ()</td>
<td>return true if input is available</td>
</tr>
<tr>
<td>virtual <code>Util::String</code> <code>GetInput</code> ()</td>
<td>read available input</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>bool <code>IsOpen</code> () const</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>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)</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>
<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>
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]

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::Win32CriticalSection
Win32::Win32CriticalSection Class Reference

#include <win32criticalsection.h>

Inheritance diagram for Win32::Win32CriticalSection:
Detailed Description

Win32-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>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Win32CriticalSection()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~Win32CriticalSection()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void Enter()</code></td>
<td>enter the critical section</td>
</tr>
<tr>
<td><code>void Leave()</code></td>
<td>leave the critical section</td>
</tr>
</tbody>
</table>
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>get adapter information</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> const</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> const</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> const</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> const</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> const</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> const</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> const</td>
<td>get always-on-top behaviour</td>
</tr>
</tbody>
</table>
void SetVerticalSyncEnabled (bool b)
turn vertical sync on/off

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 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 string

bool IsInstanceOf (const Util::FourCC &classFourCC) 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>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

<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>OnMouseButtonDown()</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>Class</td>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Input::Key::Code</td>
<td>TranslateKeyCode (WPARAM wParam) const</td>
<td>translate a Windows virtual key code into a Nebula3 key code</td>
</tr>
<tr>
<td>virtual CoreGraphics::DisplayMode</td>
<td>ComputeAdjustedWindowRect ()</td>
<td>adjust window size taking client area into account</td>
</tr>
<tr>
<td>Math::float2</td>
<td>ComputeAbsMousePos (LPARAM lParam) const</td>
<td>compute absolute mouse position from lParam</td>
</tr>
<tr>
<td>Math::float2</td>
<td>ComputeNormMousePos (const Math::float2 &amp;absMousePos) const</td>
<td>compute normalized mouse position from absolute mouse pos</td>
</tr>
<tr>
<td>bool</td>
<td>NotifyEventHandlers (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>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() [protected, virtual]

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.

```cpp
bool Base::DisplayDeviceBase::AdapterExists ( CoreGraphics::Adapter::Code adapter ) [inherited]
```

return true if adapter exists

Checks if the given adapter exists.

Reimplemented in **Direct3D9::D3D9DisplayDevice**.

```cpp
Util::Array< DisplayMode >
Base::DisplayDeviceBase::GetAvailableDisplayModes ( CoreGraphics::Adapter::Code adapter,
                            CoreGraphics::PixelFormat::Code pixelFormat )
```

get available display modes on given adapter

Returns the display modes on the given adapter in the given pixel format.

Reimplemented in **Direct3D9::D3D9DisplayDevice**.

```cpp
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 ) [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 ) [inh]

attach a display event handler

Attach an event handler to the display device.
remove a display event handler

Remove an event handler from the display device.

notify event handlers about an event

Notify all event handlers about an event.

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.
Win32::Win32Event
Win32::Win32Event Class Reference

#include <win32event.h>

Inheritance diagram for Win32::Win32Event:
Detailed Description

Win32 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><code>Win32Event ()</code></td>
<td>constructor</td>
</tr>
<tr>
<td><code>~Win32Event ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>void Signal ()</code></td>
<td>signal the event</td>
</tr>
<tr>
<td><code>void Wait () const</code></td>
<td>wait for the event to become signalled</td>
</tr>
<tr>
<td><code>bool WaitTimeout (int ms) const</code></td>
<td>wait for the event with timeout in millisecs</td>
</tr>
<tr>
<td><code>bool Peek () const</code></td>
<td>check if event is signalled</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
bool Win32::Win32Event::WaitTimeout(int timeoutInMillis) 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.
```

```cpp
bool Win32::Win32Event::Peek() const [inline]

check if event is signalled

This checks if the event is signalled and returns immediately.
```
Win32::Win32FileTime
Win32::Win32FileTime Class Reference

#include <win32filetime.h>

Inheritance diagram for Win32::Win32FileTime:

```
Win32::Win32FileTime

IO::FileTime
```
Detailed Description

Implements a Win32-specific file-access time stamp.

(C) 2006 Radon Labs GmbH
Public Member Functions

`Win32FileTime()`
constructor
<table>
<thead>
<tr>
<th>bool</th>
<th>operator== (const Win32FileTime &amp;a, const Win32FileTime &amp;b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>operator!= (const Win32FileTime &amp;a, const Win32FileTime &amp;b)</td>
</tr>
<tr>
<td>bool</td>
<td>operator&gt; (const Win32FileTime &amp;a, const Win32FileTime &amp;b)</td>
</tr>
<tr>
<td>bool</td>
<td>operator&lt; (const Win32FileTime &amp;a, const Win32FileTime &amp;b)</td>
</tr>
</tbody>
</table>
Win32::Win2FSWrapper
Win32::Win32FSWrapper Class Reference

#include <win32fswrapper.h>

Inheritance diagram for Win32::Win32FSWrapper:
Detailed Description

Internal filesystem wrapper for Win32. All paths must be native Win32 paths.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Handle OpenFile</td>
<td>(const Util::String &amp;path, IO::Stream::AccessMode accessMode,</td>
</tr>
<tr>
<td></td>
<td>IO::Stream::AccessPattern accessPattern)</td>
</tr>
<tr>
<td></td>
<td>open a file</td>
</tr>
<tr>
<td>static void CloseFile</td>
<td>(Handle h)</td>
</tr>
<tr>
<td></td>
<td>close a file</td>
</tr>
<tr>
<td>static void Write</td>
<td>(Handle h, const void *buf, IO::Stream::Size numBytes)</td>
</tr>
<tr>
<td></td>
<td>write to a file</td>
</tr>
<tr>
<td>static IO::Stream::Size</td>
<td>Read (Handle h, void *buf, IO::Stream::Size numBytes)</td>
</tr>
<tr>
<td></td>
<td>read from a file</td>
</tr>
<tr>
<td>static void Seek</td>
<td>(Handle h, IO::Stream::Offset offset, IO::Stream::SeekOrigin orig)</td>
</tr>
<tr>
<td></td>
<td>seek in a file</td>
</tr>
<tr>
<td>static IO::Stream::Position</td>
<td>Tell (Handle h)</td>
</tr>
<tr>
<td></td>
<td>get position in file</td>
</tr>
<tr>
<td>static void Flush</td>
<td>(Handle h)</td>
</tr>
<tr>
<td></td>
<td>flush a file</td>
</tr>
<tr>
<td>static bool Eof</td>
<td>(Handle h)</td>
</tr>
<tr>
<td></td>
<td>return true if at end-of-file</td>
</tr>
<tr>
<td>static IO::Stream::Size</td>
<td>GetFileSize (Handle h)</td>
</tr>
<tr>
<td></td>
<td>get size of a file in bytes</td>
</tr>
<tr>
<td>static void SetReadOnly</td>
<td>(const Util::String &amp;path, bool readOnly)</td>
</tr>
<tr>
<td></td>
<td>set read-only status of a file</td>
</tr>
<tr>
<td>static bool IsReadOnly</td>
<td>(const Util::String &amp;path)</td>
</tr>
<tr>
<td></td>
<td>get read-only status of a file</td>
</tr>
<tr>
<td>static bool DeleteFile</td>
<td>(const Util::String &amp;path)</td>
</tr>
<tr>
<td></td>
<td>delete a file</td>
</tr>
<tr>
<td>static bool DeleteDirectory</td>
<td>(const Util::String &amp;path)</td>
</tr>
<tr>
<td></td>
<td>delete an empty directory</td>
</tr>
<tr>
<td>static bool FileExists</td>
<td>(const Util::String &amp;path)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>return true if a file exists</code></td>
<td></td>
</tr>
<tr>
<td>static bool DirectoryExists</td>
<td>(const Util::String &amp;path) return true if a directory exists</td>
</tr>
<tr>
<td>static IO::FileTime</td>
<td>GetFileWriteTime (const Util::String &amp;path) get the last write-access time stamp of a file</td>
</tr>
<tr>
<td>static bool CreateDirectory</td>
<td>(const Util::String &amp;path) create a directory</td>
</tr>
<tr>
<td>static Util::Array<a href="">Util::String</a></td>
<td>ListFiles (const Util::String &amp;dirPath, const Util::String &amp;pattern) list all files in a directory</td>
</tr>
<tr>
<td>static Util::Array<a href="">Util::String</a></td>
<td>ListDirectories (const Util::String &amp;dirPath, const Util::String &amp;pattern) list all subdirectories in a directory</td>
</tr>
<tr>
<td>static Util::String</td>
<td>GetUserDirectory () get path to the current user's home directory (for user: standard assign)</td>
</tr>
<tr>
<td>static Util::String</td>
<td>GetTempDirectory () get path to the current user's temp directory (for temp: standard assign)</td>
</tr>
<tr>
<td>static Util::String</td>
<td>GetHomeDirectory () get path to the current application directory (for home: standard assign)</td>
</tr>
<tr>
<td>static Util::String</td>
<td>GetBinDirectory () get path to the current bin directory (for bin: standard assign)</td>
</tr>
</tbody>
</table>
| static bool IsDeviceName       | (const Util::String &str) return true when the string is a device name (e.g. "C:"
Member Function Documentation

Win32FSWrapper::Handle

Win32::Win32FSWrapper::OpenFile ( const Util::String & path,
    IO::Stream::AccessMode accessMode,
    IO::Stream::AccessPattern accessPattern
) [static]

open a file

Open a file using the **Win32** function CreateFile(). Returns a handle to the file which must be passed to the other **Win32FSWrapper** file methods. If opening the file fails, the function will return 0. The filename must be a native **Win32** path (no assigns, etc...).

void

Win32::Win32FSWrapper::CloseFile ( Handle handle ) [static]

close a file

Closes a file opened by **Win32FSWrapper::OpenFile()**.

Stream::Position

Win32::Win32FSWrapper::Tell ( Handle handle ) [static]

get position in file

Get current position in file.

void

Win32::Win32FSWrapper::Flush ( Handle handle ) [static]

flush a file

Flush unwritten data to file.

bool

Win32::Win32FSWrapper::Eof ( Handle handle ) [static]

return true if at end-of-file
Returns true if current position is at end of file.

Stream::Size
Win32::Win32FSWrapper::GetFileSize (Handle handle) [static]

get size of a file in bytes

Returns the size of a file in bytes.

void
Win32::Win32FSWrapper::SetReadOnly (Util::String path, &
    bool readOnly [static]

set read-only status of a file

Set the read-only status of a file.

bool
Win32::Win32FSWrapper::IsReadOnly (Util::String path) [static]

get read-only status of a file

Get the read-only status of a file.

bool
Win32::Win32FSWrapper::DeleteFile (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
Win32::Win32FSWrapper::DeleteDirectory (Util::String path) [static]

delete an empty directory

Delete an empty directory. Returns true if the operation was
successful.

```cpp
bool Win32::Win32FSWrapper::FileExists(const Util::String path) [static]
return true if a file exists
Return true if a file exists.

bool Win32::Win32FSWrapper::DirectoryExists(const Util::String path) [static]
return true if a directory exists
Return true if a directory exists.

FileTime Win32::Win32FSWrapper::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.

bool Win32::Win32FSWrapper::CreateDirectory(const Util::String path) [static]
create a directory
Creates a new directory.

Array<String> Win32::Win32FSWrapper::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.

```cpp
Array<String> Win32::Win32FSWrapper::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 ".".

```cpp
String Win32::Win32FSWrapper::GetUserDirectory()
    [static]
```

get path to the current user's home directory (for user: standard assign)

This method should return the path to the current user's home directory. This is the directory where application can write their data to. Under windows, this is the "My Files" directory.

```cpp
String Win32::Win32FSWrapper::GetTempDirectory()
    [static]
```

get path to the current user's temp directory (for temp: standard assign)

This method should return a directory for temporary files with read/write access for the current user.

```cpp
String Win32::Win32FSWrapper::GetHomeDirectory()
    [static]
```

get path to the current application directory (for home: standard assign)

This method should return the installation directory of the application. Under Nebula3, this is either the directory where the executable is
located, or 2 levels above the executable (if it is in home:bin/win32).

**String**

`Win32::Win32FSWrapper::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.
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

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>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void *</td>
<td><code>operator new (size_t size)</code></td>
<td>overloaded <code>operator new</code></td>
</tr>
<tr>
<td>void</td>
<td><code>operator delete (void *p)</code></td>
<td>overloaded <code>operator delete</code></td>
</tr>
<tr>
<td><code>Win32Guid()</code></td>
<td></td>
<td>constructor</td>
</tr>
<tr>
<td><code>Win32Guid</code></td>
<td><code>(const Win32Guid &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>Win32Guid</code></td>
<td><code>(const unsigned char *ptr, SizeT size)</code></td>
<td>construct from raw binary data as returned by <code>AsBinary()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>operator= (const Win32Guid &amp;rhs)</code></td>
<td>assignment operator</td>
</tr>
<tr>
<td>void</td>
<td><code>operator= (const Util::String &amp;rhs)</code></td>
<td>assignment operator from string</td>
</tr>
<tr>
<td>bool</td>
<td><code>operator== (const Win32Guid &amp;rhs)</code></td>
<td>const equality operator</td>
</tr>
<tr>
<td>bool</td>
<td><code>operator!= (const Win32Guid &amp;rhs)</code></td>
<td>const inequality operator</td>
</tr>
<tr>
<td>bool</td>
<td><code>operator&lt; (const Win32Guid &amp;rhs)</code></td>
<td>const less-than operator</td>
</tr>
<tr>
<td>bool</td>
<td><code>operator&lt;= (const Win32Guid &amp;rhs)</code></td>
<td>const less-or-equal operator</td>
</tr>
<tr>
<td>bool</td>
<td><code>operator&gt; (const Win32Guid &amp;rhs)</code></td>
<td>const greater-than operator</td>
</tr>
<tr>
<td>bool</td>
<td><code>operator&gt;= (const Win32Guid &amp;rhs)</code></td>
<td>const greater-or-equal operator</td>
</tr>
<tr>
<td>bool</td>
<td><code>IsValid ()</code></td>
<td>const return true if the contained guid is valid (not NIL)</td>
</tr>
<tr>
<td>void</td>
<td><code>Generate ()</code></td>
<td>generate a new guid</td>
</tr>
<tr>
<td><code>Util::String</code></td>
<td><code>AsString ()</code></td>
<td>const get as string</td>
</tr>
<tr>
<td><code>SizeT</code></td>
<td><code>AsBinary (const unsigned char * &amp;outPtr)</code></td>
<td>const get as binary data as returned by <code>AsBinary()</code></td>
</tr>
<tr>
<td>IndexT</td>
<td><strong>HashCode</strong> () const</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td>get a hash code (compatible with <em>Util::HashTable</em>)</td>
<td></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 void</td>
<td><strong>Setup()</strong></td>
<td><em>static setup method, called by Util::Setup</em></td>
</tr>
<tr>
<td>static Win32Guid</td>
<td><strong>FromString</strong> (const Util::String &amp;str)</td>
<td><em>construct from string representation</em></td>
</tr>
<tr>
<td>static Win32Guid</td>
<td><strong>FromBinary</strong> (const unsigned char *ptr, SizeT numBytes)</td>
<td><em>construct from binary representation</em></td>
</tr>
</tbody>
</table>
**Member Function Documentation**

`Win32Guid Win32::Win32Guid::FromBinary` (const unsigned *ptr, SizeT numBytes) [static]

construct from binary representation

Constructs the guid from binary data, as returned by the `AsBinary()`.

`SizeT Win32::Win32Guid::AsBinary` (const unsigned 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.

`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`. 
Win32::Win32Heap
Win32::Win32Heap Class Reference

#include <win32heap.h>
Detailed Description

Win32 implementation of the class Memory::Heap using the Win32-Heap functions. Generally switches on the Low-Fragmentation-Heap, since this seems generally suitable for most C++ applications.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Win32Heap (const char *name)</code></td>
<td>constructor <em>(name must be static string!)</em></td>
</tr>
<tr>
<td><code>~Win32Heap ()</code></td>
<td>destructor</td>
</tr>
<tr>
<td><code>const char * GetName () const</code></td>
<td>get heap name</td>
</tr>
<tr>
<td><code>void * Alloc (size_t size)</code></td>
<td>allocate a block of memory from the heap</td>
</tr>
<tr>
<td><code>void * Realloc (void *ptr, size_t newSize)</code></td>
<td>re-allocate a block of memory</td>
</tr>
<tr>
<td><code>void Free (void *ptr)</code></td>
<td>free a block of memory which has been allocated from this heap</td>
</tr>
</tbody>
</table>
Static Public Member Functions

static void Setup ()

*static setup method (called by Util::Setup)*
Member Function Documentation

void Win32::Win32Heap::Setup( ) [static]

static setup method (called by Util::Setup)

This method must be called at the beginning of the application because any threads are spawned (usually called by Util::Setup()).
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>virtual bool HandleEvent</td>
<td>(const CoreGraphics::DisplayEvent &amp;event) called when an event happens</td>
</tr>
<tr>
<td>virtual bool PutEvent</td>
<td>(const DisplayEvent &amp;event) called by DisplayDevice when an event happens</td>
</tr>
<tr>
<td>void HandlePendingEvents</td>
<td>() handle all pending events (called by consumer thread)</td>
</tr>
<tr>
<td>virtual void OnAttach</td>
<td>() called when the event handler is attached to the DisplayDevice</td>
</tr>
<tr>
<td>virtual void OnRemove</td>
<td>() called when the event handler is removed from the DisplayDevice</td>
</tr>
<tr>
<td>int GetRefCount</td>
<td>() const get the current refcount</td>
</tr>
<tr>
<td>void AddRef</td>
<td>()crement 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</td>
<td>(const Rtti &amp;rtti) const return true if this object is instance of given class</td>
</tr>
<tr>
<td>bool IsInstanceOf</td>
<td>(const Util::String &amp;className) const return true if this object is instance of given class by string</td>
</tr>
<tr>
<td>bool IsInstanceOf</td>
<td>(const Util::FourCC &amp;classFourCC) const return true if this object is instance of given class by fourcc</td>
</tr>
<tr>
<td>bool IsA</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>bool IsA</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>bool IsA</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 Util::String &amp;</td>
<td>GetClassName() const</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 <em>FourCC</em> code</td>
<td></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

```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.

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::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
**Public Member Functions**

<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>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>IsOpen()</code> const</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> const</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> const</td>
<td>get the default mouse input handler</td>
</tr>
<tr>
<td><code>GetDefaultGamePad(IndexT playerIndex)</code> const</td>
<td>get default gamepad handler (up to 4)</td>
</tr>
<tr>
<td><code>AttachInputHandler(Input::InputPriority::Code pri, const Ptr&lt;Input::InputHandler&gt; &amp;inputHandler)</code></td>
<td>attach an input handler</td>
</tr>
</tbody>
</table>
| `RemoveInputHandler(const Ptr<Input::InputHandler> &inputHandler)` | }
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

void ReleaseKeyboardCapture (const Ptr < Input::InputHandler > &inputHandler)
only call from InputHandler: release keyboard capture

int GetRefCount () const
get the current refcount

void AddRef ()
increment refcount by one
<table>
<thead>
<tr>
<th>void</th>
<th><strong>Release ()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>decrement refcount and destroy object if refcount is zero</td>
</tr>
<tr>
<td>bool</td>
<td><strong>IsInstanceOf (const Rtti &amp;rtti) const</strong></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 (const Util::String &amp;className) const</strong></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 (const Util::FourCC &amp;classFourCC) const</strong></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 (const Rtti &amp;rtti) const</strong></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 (const Util::String &amp;rttiName) const</strong></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 (const Util::FourCC &amp;rttiFourCC) const</strong></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 () const</strong></td>
</tr>
<tr>
<td></td>
<td>get the class name</td>
</tr>
<tr>
<td>Util::FourCC</td>
<td><strong>GetClassFourCC () const</strong></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>bool OpenDInputMouse</td>
<td>setup the DirectInput mouse device for tracking mouse movement</td>
</tr>
<tr>
<td>void CloseDInputMouse</td>
<td>shutdown the DirectInput mouse device</td>
</tr>
<tr>
<td>void ReadDInputMouse</td>
<td>get mouse readings</td>
</tr>
<tr>
<td>const Math::float2 &amp; GetMouseMovement</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::EndFrame ( ) [inherited]

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 ) [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!
void Base::InputServerBase::ClearMouseCapture() [inherited]
clear the current mouse capture (if exists)
This clears the currently set mouse capture (if exists).

void Base::InputServerBase::ClearKeyboardCapture() [inherited]
clear the current keyboard capture (if exists)
This clears the currently set keyboard capture (if exists).

void Base::InputServerBase::ClearCapture() [inherited]
clear both mouse and keyboard captures
This clears the mouse and keyboards captures, if set.

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.

void Base::InputServerBase::ReleaseMouseCapture(const Ptr<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(constPtr<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.

```cpp
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]
```

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.

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

dump 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::Win32Interlocked
Win32::Win32Interlocked Class Reference

#include <win32interlocked.h>

Inheritance diagram for Win32::Win32Interlocked:
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>Increment</td>
<td>(int volatile &amp;var)</td>
</tr>
<tr>
<td>Decrement</td>
<td>(int volatile &amp;var)</td>
</tr>
<tr>
<td>Add</td>
<td>(int volatile &amp;var, int add)</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by *doxygen* at Tue Feb 19 12:16:54 2008
Win32::Win32IpAddress
Win32::Win32IpAddress Class Reference

#include <win32ipaddress.h>
Detailed Description

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) 2006 Radon Labs GmbH
### Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Win32IpAddress()</code></td>
<td>default constructor</td>
</tr>
<tr>
<td><code>Win32IpAddress(const Win32IpAddress &amp;rhs)</code></td>
<td>copy constructor</td>
</tr>
<tr>
<td><code>Win32IpAddress(const IO::URI &amp;uri)</code></td>
<td>construct from URI</td>
</tr>
<tr>
<td><code>Win32IpAddress(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 Win32IpAddress &amp;rhs)</code></td>
<td>const equality operator</td>
</tr>
<tr>
<td>bool <code>operator&lt;(const Win32IpAddress &amp;rhs)</code></td>
<td>const less-than operator</td>
</tr>
<tr>
<td>bool <code>operator&gt;(const Win32IpAddress &amp;rhs)</code></td>
<td>const 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></td>
<td>const 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></td>
<td>const get port number</td>
</tr>
<tr>
<td>const Util::String &amp; <code>GetHostAddr()</code></td>
<td>const get the ip address resulting from the host name as string</td>
</tr>
</tbody>
</table>
Protected Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SetSockAddr</code> (const sockaddr_in &amp;addr)</td>
<td>set sockaddr_in directly</td>
</tr>
<tr>
<td><code>const sockaddr_in &amp; GetSockAddr () const</code></td>
<td>get sockaddr_in field</td>
</tr>
</tbody>
</table>
### Static Protected Member Functions

<table>
<thead>
<tr>
<th>static bool</th>
<th>GetHostByName (const Util::String &amp;hostName, in_addr &amp;outAddr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>perform address resolution, understands special host names</em></td>
</tr>
<tr>
<td>static bool</td>
<td>IsInetAddr (const in_addr *addr)</td>
</tr>
<tr>
<td></td>
<td><em>return true if an address is an internet address (not class A,B,C)</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
void Win32::Win32IpAddress::ExtractFromUri ( const 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".

```cpp
void Win32::Win32IpAddress::SetHostName ( const 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().

```cpp
const String & Win32::Win32IpAddress::GetHostName ( ) const

get host name

Get the host name.

```cpp
void Win32::Win32IpAddress::SetPort ( ushort port )

set port number

Set the port number. Will be translated to network byte order internally.

```cpp
ushort Win32::Win32IpAddress::GetPort ( ) const
```
get port number

Get the port number in host byte order.

```cpp
const String &
Win32::Win32IpAddress::GetHostAddr()
```

get the ip address resulting from the host name as string

Return the in address as string.

```cpp
void
Win32::Win32IpAddress::SetSockAddr(const sockaddr_in & sa)
```

set sockaddr_in directly

Set the address directly from a sockaddr_in struct. This will set the host name to the string representation of the host address.

```cpp
const sockaddr_in &
Win32::Win32IpAddress::GetSockAddr()
```

get sockaddr_in field

Get the sockaddr_in struct, which has either been set directly with `SetSockAddr()` or indirectly through host name, port number or from an URI.

```cpp
bool
Win32::Win32IpAddress::GetHostByName(const Util::String hostName, & in_addr & outAddr)
```

perform address resolution, understands special host names

This resolves a host name into a IPv4 ip address. The ip address is returned in network byte order in the hostAddress argument. The return value indicates whether the operation was successful. The following special hostnames can be defined:
- "any" resolves to INADDR_ANY (0.0.0.0)
- "broadcast" resolves to INADDR_BROADCAST (255.255.255.255)
- "localhost" resolves to 127.0.0.1
- "self" resolves to the first address of this host
- "inetself" resolves to the first address which is not a LAN address

An empty host name is invalid. A hostname can also be an address string of the form xxx.yyy.zzz.www.

```cpp
bool Win32::Win32IpAddress::IsInetAddr (const in_addr *addr) [static, protected]
```

return true if an address is an internet address (not class A,B,C)

This method checks if the provided address is an "internet" address, not a LAN address (not a class A, B or C network address).
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:
Detailed Description

Win32 implementation of MiniDump.

(C) 2007 Radon Labs GmbH
Static Public Member Functions

static bool WriteMiniDump ()

write a mini dump
Member Function Documentation

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.
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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{const Math::float2 &amp; GetMovement()} const</td>
<td>get mouse movement</td>
</tr>
<tr>
<td>\texttt{virtual void BeginCapture()}</td>
<td>capture input to this event handler</td>
</tr>
<tr>
<td>\texttt{virtual void EndCapture()}</td>
<td>end input capturing to this event handler</td>
</tr>
<tr>
<td>\texttt{bool ButtonPressed(Input::MouseButton::Code btn) const}</td>
<td>return true if button is currently pressed</td>
</tr>
<tr>
<td>\texttt{bool ButtonDown(Input::MouseButton::Code btn) const}</td>
<td>return true if button was down at least once in current frame</td>
</tr>
<tr>
<td>\texttt{bool ButtonUp(Input::MouseButton::Code btn) const}</td>
<td>return true if button was up at least once in current frame</td>
</tr>
<tr>
<td>\texttt{bool ButtonDoubleClicked(Input::MouseButton::Code btn) const}</td>
<td>return true if a button has been double clicked</td>
</tr>
<tr>
<td>\texttt{bool WheelForward()} const</td>
<td>return true if mouse wheel rotated forward</td>
</tr>
<tr>
<td>\texttt{bool WheelBackward()} const</td>
<td>return true if mouse wheel rotated backward</td>
</tr>
<tr>
<td>\texttt{const Math::float2 &amp; GetPixelPosition()} const</td>
<td>get current absolute mouse position (in pixels)</td>
</tr>
<tr>
<td>\texttt{const Math::float2 &amp; GetScreenPosition()} const</td>
<td>get current screen space mouse position (0.0 .. 1.0)</td>
</tr>
<tr>
<td>\texttt{bool IsAttached()} const</td>
<td>return true if the input handler is currently attached</td>
</tr>
<tr>
<td>\texttt{bool IsCapturing()} const</td>
<td>return true if this input handler captures input</td>
</tr>
<tr>
<td>\texttt{int GetRefCount()} const</td>
<td>get the current refcount</td>
</tr>
<tr>
<td>\texttt{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><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></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></th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void</td>
<td><strong>OnAttach</strong> ()</td>
<td>called when the handler is attached to the input server</td>
</tr>
<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**

```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.
Win32::Win32Socket
Win32::Win32Socket Class Reference

#include <win32socket.h>

Inheritance diagram for Win32::Win32Socket:

- Core::RefCounted
- Win32::Win32Socket
- Net::Socket
Detailed Description

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) 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><strong>Win32Socket ()</strong></td>
<td>constructor</td>
</tr>
<tr>
<td><strong>~Win32Socket ()</strong></td>
<td>destructor</td>
</tr>
<tr>
<td><strong>Open (Protocol p)</strong></td>
<td>open the socket</td>
</tr>
<tr>
<td><strong>Close ()</strong></td>
<td>close the socket</td>
</tr>
<tr>
<td><strong>IsOpen () const</strong></td>
<td>return true if the socket is open</td>
</tr>
<tr>
<td><strong>GetErrorCode () const</strong></td>
<td>get the last error code</td>
</tr>
<tr>
<td><strong>GetErrorString () const</strong></td>
<td>get the last error string</td>
</tr>
<tr>
<td><strong>SetAddress (const Net::IpAddress &amp;a)</strong></td>
<td>set internet address of socket</td>
</tr>
<tr>
<td><strong>GetAddress () const</strong></td>
<td>get internet address of socket</td>
</tr>
<tr>
<td><strong>SetBroadcast (bool b)</strong></td>
<td>set the broadcast flag (SO_BROADCAST)</td>
</tr>
<tr>
<td><strong>GetBroadcast ()</strong></td>
<td>get the broadcast flag</td>
</tr>
<tr>
<td><strong>SetDontLinger (bool b)</strong></td>
<td>set the don't linger flag (SO_DONTLINGER)</td>
</tr>
<tr>
<td><strong>GetDontLinger ()</strong></td>
<td>get the don't linger flag</td>
</tr>
<tr>
<td><strong>SetKeepAlive (bool b)</strong></td>
<td>set the keepalive flag (SO_KEEPALIVE)</td>
</tr>
<tr>
<td><strong>GetKeepAlive ()</strong></td>
<td>get the keepalive flag</td>
</tr>
<tr>
<td><strong>SetReUseAddr (bool b)</strong></td>
<td>set reuseaddr flag (SO_REUSEADDR)</td>
</tr>
<tr>
<td><strong>GetReUseAddr ()</strong></td>
<td></td>
</tr>
</tbody>
</table>
get reuseaddr flag

void SetNoDelay (bool b)
set nodelay flag (TCP_NODELAY)

bool GetNoDelay ()
get nodelay flag

void SetRecvBufSize (SizeT s)
set receive buffer size

SizeT GetRecvBufSize ()
get receive buffer size

void SetSendBufSize (SizeT s)
set send buffer size

SizeT GetSendBufSize ()
get send buffer size

void SetBlocking (bool b)
set blocking mode (FIONBIO)

bool GetBlocking () const
get blocking mode

SizeT GetMaxMsgSize ()
get the maximum message size that can be sent atomically

bool Bind ()
bind socket to ip address

bool IsBound () const
return true if the socket is bound to an address

bool Listen ()
listen for incoming connections (for server sockets)

bool Accept (Ptr< Net::Socket > &outSocket)
accept incoming connection, return a new socket (for server sockets)

Result Connect ()
connect to the sockets address (for client sockets)

bool IsConnected ()

Result  Send (const void *buf, SizeT numBytes, SizeT &bytesSent)
send raw data into the socket

bool HasRecvData ()
return true if recv data is available at the socket
<table>
<thead>
<tr>
<th>Result</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Recv</strong> (void *buf, SizeT bufSize, SizeT &amp;bytesReceived)</td>
<td>receive raw data from the socket</td>
</tr>
<tr>
<td></td>
<td><strong>SendTo</strong> (const void *buf, SizeT numBytes, uint addr, ushort port, SizeT &amp;bytesSent)</td>
<td>send raw data to address for connectionless sockets</td>
</tr>
<tr>
<td></td>
<td><strong>RecvFrom</strong> (void *buf, SizeT bufSize, uint addr, ushort port, SizeT &amp;bytesReceived)</td>
<td>receive raw data from address for connectionless sockets</td>
</tr>
</tbody>
</table>

```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

```cpp
void Win32::Win32Socket::SetAddress (const Net::IpAddress a) [inline]
```

set internet address of socket

Set internet address of socket.

```cpp
const Net::IpAddress & Win32::Win32Socket::GetAddress () const [inline]
```

get internet address of socket

Get internet address of socket.

```cpp
void Win32::Win32Socket::SetBlocking (bool b)
```

set blocking mode (FIONBIO)

Set the socket to blocking mode.

```cpp
bool Win32::Win32Socket::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 Win32::Win32Socket::Listen ()
```

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.
bool Win32::Win32Socket::Accept (Ptr<Net::Socket outSocket> &)
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".

Win32Socket::Result Win32::Win32Socket::Connect ()
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.

bool Win32::Win32Socket::IsConnected ()
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.

Win32Socket::Result Win32::Win32Socket::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.

```cpp
bool Win32::Win32Socket::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.

```cpp
Win32Socket::Result Win32::Win32Socket::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 `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.

```cpp
Win32Socket::Result Win32::Win32Socket::SendTo(const void * buf,
                                              const void * buf,
                                              SizeT bufSize)
```
send raw data to address for connectionless sockets

FIXME: this is the send method for connectionless sockets using the UDP protocol.

```
Win32Socket::Result
Win32::Win32Socket::RecvFrom ( void * buf,
    SizeT bufSize,
    uint addr,
    ushort port,
    SizeT & bytesReceived
)
```

receive raw data from address for connectionless sockets

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.

```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.
Win32::Win32Thread
Win32::Win32Thread Class Reference

#include <win32thread.h>

Inheritance diagram for Win32::Win32Thread:
Detailed Description

Win32 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>Win32Thread()</td>
<td>Constructor</td>
</tr>
<tr>
<td>virtual ~Win32Thread()</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 (unsigned int s)</td>
<td>set stack size in bytes (default is 4 KByte)</td>
</tr>
<tr>
<td>unsigned int 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 <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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static const char * GetMyThreadName()</code></td>
<td>obtain name of thread from within thread code</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>virtual void <strong>EmitWakeupSignal</strong> ()</td>
<td><em>override this method if your thread loop needs a wakeup call before stopping</em></td>
</tr>
<tr>
<td>virtual void <strong>DoWork</strong> ()</td>
<td><em>this method runs in the thread context</em></td>
</tr>
<tr>
<td>bool <strong>ThreadStopRequested</strong> () const</td>
<td><em>check if stop is requested, call from <strong>DoWork()</strong> to see if the thread proc should quit</em></td>
</tr>
</tbody>
</table>
Member Function Documentation

void Win32::Win32Thread::SetName ( const Util::String & 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().

const Util::String & Win32::Win32Thread::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().

void Win32::Win32Thread::Start ( )

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 returns immediately without waiting for the thread to start.

void Win32::Win32Thread::Stop ( )

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.

```cpp
const char *
Win32::Win32Thread::GetMyThreadName() [static]
```

obtain name of thread from within thread code

Static method to obtain the current thread name from anywhere in the thread's code.

```cpp
void
Win32::Win32Thread::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.

```cpp
void
Win32::Win32Thread::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.

```cpp
bool
Win32::Win32Thread::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.
Win32::Win32Timer
Win32::Win32Timer Class Reference

#include <win32timer.h>

Inheritance diagram for Win32::Win32Timer:
Detailed Description

Win32 implementation of the Time::Timer class. Under Win32, time measurement uses the QueryPerformanceCounter() methods.

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><code>Win32Timer()</code></td>
<td>Constructor</td>
</tr>
<tr>
<td><code>Start()</code></td>
<td>Start/continue the timer</td>
</tr>
<tr>
<td><code>Stop()</code></td>
<td>Stop the timer</td>
</tr>
<tr>
<td><code>Reset()</code></td>
<td>Reset the timer</td>
</tr>
<tr>
<td><code>Running()</code> const</td>
<td>Return true if currently running</td>
</tr>
<tr>
<td><code>GetTime()</code> const</td>
<td>Get current time in seconds</td>
</tr>
<tr>
<td><code>GetTicks()</code> const</td>
<td>Get current time in ticks</td>
</tr>
</tbody>
</table>
Member Function Documentation

void Win32::Win32Timer::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 Win32::Win32Timer::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 Win32::Win32Timer::Reset()
reset the timer
Reset the timer so that will start counting at zero again.

bool Win32::Win32Timer::Running() const
return true if currently running
Returns true if the timer is currently running.

Timing::Time Win32::Win32Timer::GetTime() const
get current time in seconds
This returns the timer's current time in seconds.

```cpp
uint
Win32::Win32Timer::GetTicks() const
```

get current time in ticks

This returns the timer's current time in "ticks".
XInput::XInputGamePad
#include <xinputgamepad.h>

Inheritance diagram for XInput::XInputGamePad:

```
Core::RefCounted

Input::InputHandler

Base::GamePadBase

XInput::XInputGamePad

Input::GamePad
```
Detailed Description

Common gamepad support for Xbox360 and Windows.

(C) 2007 Radon Labs GmbH
Public Types

enum Button
gamepad buttons
**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>virtual <code>~XInputGamePad()</code></td>
<td>destructor</td>
</tr>
<tr>
<td>bool <code>IsConnected()</code></td>
<td>return true if this game pad is currently connected</td>
</tr>
<tr>
<td>IndexT <code>GetPlayerIndex()</code></td>
<td>get the player index of this game pad</td>
</tr>
<tr>
<td>bool <code>ButtonPressed(Button btn)</code></td>
<td>return true if a button is currently pressed</td>
</tr>
<tr>
<td>bool <code>ButtonDown(Button btn)</code></td>
<td>return true if button was down at least once in current frame</td>
</tr>
<tr>
<td>bool <code>ButtonUp(Button btn)</code></td>
<td>return true if button was up at least once in current frame</td>
</tr>
<tr>
<td>float <code>GetAxisValue(Axis axis)</code></td>
<td>get current axis value</td>
</tr>
<tr>
<td>void <code>SetLowFrequencyVibrator(float f)</code></td>
<td>set low-frequency vibration effect (0.0f .. 1.0f)</td>
</tr>
<tr>
<td>float <code>GetLowFrequencyVibrator()</code></td>
<td>get low-frequency vibration</td>
</tr>
<tr>
<td>void <code>SetHighFrequencyVibrator(float f)</code></td>
<td>set high-frequency vibration effect (0.0f .. 1.0f)</td>
</tr>
<tr>
<td>float <code>GetHighFrequencyVibrator()</code></td>
<td>get high-frequency vibration</td>
</tr>
<tr>
<td>bool <code>IsAttached()</code></td>
<td>return true if the input handler is currently attached</td>
</tr>
<tr>
<td>virtual void <code>BeginCapture()</code></td>
<td>capture input to this event handler</td>
</tr>
<tr>
<td>virtual void <code>EndCapture()</code></td>
<td>end input capturing to this event handler</td>
</tr>
<tr>
<td>bool <code>IsCapturing()</code></td>
<td>return true if this input handler captures input</td>
</tr>
<tr>
<td>int <code>GetRefCount()</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>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 <code>OnBeginFrame()</code></td>
<td>called on <code>InputServer::BeginFrame()</code></td>
</tr>
<tr>
<td>void <code>UpdateButtonState</code> (const XINPUT_GAMEPAD &amp;curState, WORD xiBtn, Button btn)</td>
<td>update the state of a game pad button</td>
</tr>
<tr>
<td>void <code>UpdateTriggerAxis</code> (const XINPUT_GAMEPAD &amp;curState, Axis axis)</td>
<td>update the state of a trigger axis</td>
</tr>
<tr>
<td>void <code>UpdateThumbAxis</code> (const XINPUT_GAMEPAD &amp;curState, Axis axis)</td>
<td>update the state of a thumb stick axis</td>
</tr>
<tr>
<td>virtual void <code>OnAttach()</code></td>
<td>called when the handler is attached to the input server</td>
</tr>
<tr>
<td>virtual void <code>OnReset()</code></td>
<td>reset the input handler</td>
</tr>
<tr>
<td>virtual void <code>OnRemove()</code></td>
<td>called when the handler is removed from the input server</td>
</tr>
<tr>
<td>virtual void <code>OnEndFrame()</code></td>
<td>called on <code>InputServer::EndFrame()</code></td>
</tr>
<tr>
<td>virtual void <code>OnObtainCapture()</code></td>
<td>called when input handler obtains capture</td>
</tr>
<tr>
<td>virtual void <code>OnReleaseCapture()</code></td>
<td>called when input handler looses capture</td>
</tr>
<tr>
<td>virtual bool <code>OnEvent</code> (const InputEvent &amp;inputEvent)</td>
<td>called when an input event should be processed</td>
</tr>
</tbody>
</table>
Member Function Documentation

```cpp
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.
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
Compares the previous and current state of a game pad button and
updates the parent class' state accordingly.

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`.

```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.
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
The Nebula Device 3 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- AngularPFeedbackLoop
- App::Application
  - App::AsyncRenderApplication
    - App::AsyncViewerApplication
  - App::ConsoleApplication
  - App::GameApplication
  - App::RenderApplication
    - App::ViewerApplication
- App::GameStateHandler
- Application::StateHandler
- AsyncGraphics::Handle
- AsyncHttp::AsyncHttpHandler
- AsyncHttp::HttpServerProxy
- Attr::AccessMode
- Attr::AttributeContainer
- Attr::AttributeDefinitionBase
  - Attr::AttributeDefinition<VALUETYPE, TYPE>
- Attr::AttrId
  - Attr::BlobAttrId
  - Attr::BoolAttrId
  - Attr::Float4AttrId
  - Attr::FloatAttrId
  - Attr::GuidAttrId
  - Attr::IntAttrId
  - Attr::Matrix44AttrId
  - Attr::StringAttrId
- Base::CalendarTimeBase
  - Win32::Win32CalendarTime
    - Timing::CalendarTime
- Base::D3D9TransformDevice
- Base::PreShader
- Base::TextureBase::MapInfo
- BaseGameFeature::CategoryManager::Category
- BaseGameFeature::CategoryManager::Entry
- BaseGameFeature::EntityManager::DelayedJob
- BaseGameFeature::FactoryManager::BluePrint
- BaseGameFeature::LevelLoader
- core
- Core::Factory
- Core::Ptr
- Core::RefCounted
  - AsyncGraphics::DisplayProxy
  - AsyncGraphics::GraphicsEntityProxy
    - AsyncGraphics::CameraEntityProxy
  - AsyncGraphics::GraphicsServerProxy
  - AsyncGraphics::ViewProxy
  - Attr::AttributeTable
  - Base::DisplayDeviceBase
    - Win32::Win32DisplayDevice
      - Direct3D9::D3D9DisplayDevice
        - CoreGraphics::DisplayDevice
  - Base::InputServerBase
    - Win32::Win32InputServer
      - Input::InputServer
  - 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
  - Direct3D9::D3D9ShapeRenderer
  - CoreGraphics::ShapeRenderer
- Base::TransformDeviceBase
- Base::VertexLayoutBase
  - Direct3D9::D3D9VertexLayout
  - CoreGraphics::VertexLayout
- BaseGameFeature::EntityLoaderBase
  - BaseGameFeature::EntityLoader
  - BaseGameFeature::EnvironmentLoader
- BaseGameFeature::LoaderServer
- BaseGameFeature::UserProfile
- Core::CoreServer
- CoreGraphics::DisplayEventHandler
  - CoreGraphics::ThreadSafeDisplayEventHandler
    - Win32::Win32InputDisplayEventHandler
- CoreGraphics::RenderEventHandler
  - CoreGraphics::ThreadSafeRenderEventHandler
- CoreGraphics::VertexLayoutServer
- Frame::FrameBatch
- Frame::FramePass
- Frame::FramePostEffect
- Frame::FrameServer
- Frame::FrameShader
- Game::Entity
- Game::FeatureUnit
- Game::GameServer
- Graphics::Cell
- Graphics::GraphicsEntity
  - Graphics::CameraEntity
  - Graphics::ModelEntity
    - Graphics::ActorEntity
  - Lighting::AbstractLightEntity
    - Lighting::GlobalLightEntity
    - Lighting::SpotLightEntity
- Graphics::GraphicsServer
- Graphics::Stage
- Graphics::StageBuilder
  - Graphics::QuadtreeStageBuilder
  - Graphics::SimpleStageBuilder
- Graphics::View
- Http::HttpRequestHandler
  - Debug::CorePageHandler
  - Debug::DisplayPageHandler
  - Debug::IoPageHandler
  - Debug::MemoryPageHandler
  - Debug::MeshPageHandler
  - Debug::ScriptingPageHandler
  - Debug::ShaderPageHandler
  - Debug::TexturePageHandler
  - Http::DefaultHttpRequestHandler
- Http::HttpServer
- Input::InputHandler
  - Base::GamePadBase
    - XInput::XInputGamePad
      - Input::GamePad
  - Base::KeyboardBase
    - Input::Keyboard
  - Base::MouseBase
    - Win32::Win32Mouse
      - Input::Mouse
- IO::Console
- IO::ConsoleHandler
  - Win32::Win32ConsoleHandler
- IO::IoServer
- IO::Stream
  - IO::FileStream
  - IO::MemoryStream
  - IO::ZipFileStream
- IO::StreamReader
  - Http::HttpRequestReader
  - IO::BinaryReader
  - IO::TextReader
  - IO::XmlReader
  - Messaging::MessageReader
- Models::ModelReader
  - Models::BinaryModelReader
  - Models::N2ModelReader
  - Models::XmlModelReader
- IO::StreamWriter
  - Http::Base64Writer
  - Http::HtmlPageWriter
  - Http::HttpResponseWriter
  - IO::BinaryWriter
  - IO::TextWriter
  - IO::XmlWriter
  - Messaging::MessageWriter
  - Models::ModelWriter
    - Models::BinaryModelWriter
    - Models::XmlModelWriter
- IO::ZipArchive
- IO::ZipDirEntry
- IO::ZipFileEntry
- IO::ZipFileSystem
- Lighting::LightServerBase
  - Lighting::SM30LightServer
  - Lighting::LightServer
- Messaging::AsyncPort
  - AsyncGraphics::AsyncGraphicsInterface
  - AsyncHttp::AsyncHttpContext
- Messaging::Handler
  - AsyncGraphics::AsyncGraphicsHandler
- Messaging::Message
  - BaseGameFeature::MoveFollow
  - GraphicsFeature::CameraDistance
  - GraphicsFeature::CameraFocus
  - GraphicsFeature::CameraOrbit
  - GraphicsFeature::CameraReset
  - GraphicsFeature::GetGraphicsEntities
  - GraphicsFeature::InputFocus
  - Http::HttpRequest
  - Interface::CopyFile
  - Interface::IOMessage
    - Interface::CreateDirectory
- Interface::DeleteDirectory
- Interface::DeleteFile
- Interface::MountZipArchive
- Interface::ReadStream
- Interface::WriteStream

- Messaging::Port
- Game::Property
- Messaging::Dispatcher

- Game::Manager
  - BaseGameFeature::CategoryManager
  - BaseGameFeature::EntityManager
  - BaseGameFeature::EnvEntityManager
  - BaseGameFeature::EnvQueryManager
  - BaseGameFeature::FactoryManager
  - BaseGameFeature::FocusManager
  - BaseGameFeature::GlobalAttrsManager

- Models::ModelInstance
- Models::ModelNode
  - Models::TransformNode
    - Models::CharacterNode
  - Models::StateNode
  - Models::ShapeNode
    - Models::ParticleSystemNode
    - Models::SkinShapeNode

- Models::ModelNodeInstance
  - Models::TransformNodeInstance
    - Models::CharacterNodeInstance
  - Models::StateNodeInstance
    - Models::ShapeNodeInstance
      - Models::ParticleSystemNodeInstance
      - Models::SkinShapeNodeInstance

- Models::ModelServer
- Models::VisResolver
- Net::TcpClient
- Net::TcpClientConnection
- Net::TcpServer
- Resources::ManagedResource
  - Models::ManagedModel
  - Resources::ManagedMesh
- Resources::ManagedTexture
- Resources::Resource
  - Base::MeshBase
    - CoreGraphics::Mesh
  - Base::ResourceBase
    - Base::IndexBufferBase
      - CoreGraphics::CPUIndexBuffer
      - Direct3D9::D3D9IndexBuffer
        - CoreGraphics::IndexBuffer
    - Base::TextureBase
      - Direct3D9::D3D9Texture
        - CoreGraphics::Texture
    - Base::VertexBufferBase
      - CoreGraphics::CPUVertexBuffer
      - Direct3D9::D3D9VertexBuffer
        - CoreGraphics::VertexBuffer
  - Base::ShaderBase
    - Direct3D9::D3D9Shader
      - CoreGraphics::Shader
  - Models::Model
- Resources::ResourceLoader
  - Base::MemoryIndexBufferLoaderBase
    - CoreGraphics::CPUMemoryIndexBufferLoader
    - Direct3D9::D3D9MemoryIndexBufferLoader
      - CoreGraphics::MemoryIndexBufferLoader
  - Base::MemoryVertexBufferLoaderBase
    - CoreGraphics::CPUMemoryVertexBufferLoader
    - Direct3D9::D3D9MemoryVertexBufferLoader
      - CoreGraphics::MemoryVertexBufferLoader
  - CoreGraphics::StreamAnimationLoader
  - CoreGraphics::StreamMeshLoader
  - Direct3D9::D3D9StreamShaderLoader
    - CoreGraphics::StreamShaderLoader
  - Direct3D9::D3D9StreamTextureLoader
    - CoreGraphics::StreamTextureLoader
  - Models::StreamModelLoader
  - Resources::DynamicMeshResourceLoader
- Resources::ResourceManager
- Resources::ResourceMapper
- Resources::SimpleResourceMapper
  - Resources::ResourceSaver
    - Base::StreamTextureSaverBase
      - Direct3D9::D3D9StreamTextureSaver
    - CoreGraphics::StreamTextureSaver
  - Resources::SharedResourceServer
  - Scripting::Command
  - Scripting::ScriptServer
    - Scripting::LuaServer
  - Win32::Win32Socket
    - Net::Socket
  - Win32::Win32Thread
    - Threading::Thread
- Core::Rtti
- Core::Singleton
- Core::SysFunc
- CoreGraphics::Adapter
- CoreGraphics::AdapterInfo
- CoreGraphics::AntiAliasQuality
- CoreGraphics::BatchType
- CoreGraphics::DisplayEvent
- CoreGraphics::DisplayMode
- CoreGraphics::ImageFileFormat
- CoreGraphics::IndexType
- CoreGraphics::PixelFormat
- CoreGraphics::PrimitiveGroup
- CoreGraphics::PrimitiveTopology
- CoreGraphics::RenderEvent
- CoreGraphics::ShaderFeature
- CoreGraphics::TransformDevice
- CoreGraphics::VertexComponent
- Direct3D9::D3D9Types
- Frame::FrameShaderLoader
- Frame::LightingMode
- Frame::SortingMode
- Game::BaseGameFeatureUnit
- Game::CoreFeature
- Game::PhysicsFeatureUnit
- GraphicsFeature::GraphicsFeatureUnitUnit
- GraphicsFeature::SetVisible
- Http::HtmlElement
- Http::HttpMethod
- Http::HttpStatus
- Input::InputEvent
- Input::InputPriority
- Input::Key
- Input::MouseButton
- IO::Interface
- IO::IOInterfaceHandlerBase
- IO::MediaType
- IO::URI
- Legacy::Nax2StreamReader
- Legacy::Nvx2StreamReader
- Lighting::LightType
- Lighting::PSSMUtil
- Lighting::SM30ShadowServer
  - Lighting::ShadowServer
- Math::bbox
- Math::ClipStatus
- Math::float2
- Math::float4
  - Math::point
  - Math::vector
- Math::line
- Math::matrix44
- Math::noise
- Math::plane
- Math::polar
- Math::quaternion
- Math::rectangle< TYPE >
- Math::sphere
- Math::transform44
- Memory::Heap
- Memory::Memory
- Memory::MemoryStatus
- Messaging::AddAttachment
- Messaging::AddSkin
- Messaging::AnimationHotspotTriggered
- Messaging::ApplyImpulseAtPos
- Messaging::FadeAnimation
- Messaging::GetActiveAnimation
- Messaging::GetAnimationInfo
- Messaging::GetAttachmentEntities
- Messaging::GetHotspotTime
- Messaging::GetJointMatrix
- Messaging::GetPhysicsEntity
- Messaging::HasAttachment
- Messaging::HideAttachment
- Messaging::Id
- Messaging::MoveDirection
- Messaging::MoveGoto
- Messaging::MoveRotate
- Messaging::MoveSetVelocity
- Messaging::MoveStop
- Messaging::MoveTurn
- Messaging::RemAttachment
- Messaging::RemSkin
- Messaging::SetAnimation
- Messaging::SetFadeAnimationMix
- Messaging::SetTransform
- Messaging::ShowAttachment
- Messaging::UpdateAttachments
- Messaging::UpdateTransform
- Models::ModelNodeType
- Models::VisResolveContainer< TYPE >
- Net::IpAddress
- PFeedbackLoop
- PhysicsFeature::MouseGripperProperty
- PhysicsFeature::PhysicsProperty
- PIDFeedbackLoop
- PQuatFeedbackLoop
- PreShaders::BoxFilterKernel
- PreShaders::GaussianBlur5x5FilterKernel
- QuadTree
- RenderUtil::MayaCameraUtil
- Resources::ResourceId
- Scripting::Arg
- Scripting::ArgsBlock
- System::AppEntry
- System::ByteOrder
- System::Cpu
- System::Win32Registry
- Threading::Barrier
- Threading::ThreadLocalPtr< T >
- Timing::InputTimeSource
- Timing::SystemTimeSource
- Timing::TimeManager
- Timing::TimeSource
- Timing::TimingTimeSource
- Util::Array< TYPE >
- Util::Array< Util::KeyValuePair< PRITYPE, TYPE > >
- Util::Atom< TYPE >
- Util::Blob
- Util::CharEnhancementUtil
- Util::CmdLineArgs
- Util::Crc
- Util::Dictionary< KEYTYPE, VALUETYPE >
- Util::FixedArray< TYPE >
- Util::FixedTable< TYPE >
- Util::FourCC
- Util::Guid
- Util::HashTable< KEYTYPE, VALUETYPE >
- Util::KeyValuePair< KEYTYPE, VALUETYPE >
  - Attr::Attribute
- Util::KeyValuePair< Attr::AttrId, Util::Variant >
- Util::KeyValuePair< PRITYPE, TYPE >
- Util::KeyValuePair< Util::String, Util::String >
  - IO::Assign
- Util::LightFlickerUtil
- Util::List< TYPE >
  - Core::RefCountedList
- Util::List< TYPE >::Iterator
- Util::List< Core::RefCounted * >
- Util::Proxy< TYPE >
- Util::QuadTree< TYPE >::Node
- Util::Queue< TYPE >
- Threading::SafeQueue< TYPE >
- Util::Queue< Util::KeyValuePair< PRITYPE, TYPE > >
  - Threading::SafePriorityQueue< PRITYPE, TYPE >
- Util::SegmentedGfxUtil
- Util::SimpleTree< VALUETYPE >
- Util::SimpleTree< VALUETYPE >::Node
- Util::Stack< TYPE >
- Util::String
- Util::StringAtom
- Util::Variant
- Win32::SysFunc
- Win32::Win32Barrier
- Win32::Win32CriticalSection
  - Threading::CriticalSection
- Win32::Win32Event
  - Threading::Event
- Win32::Win32FileTime
  - IO::FileTime
- Win32::Win32FSWrapper
  - Internal::FSWrapper
- Win32::Win32Guid
- Win32::Win32Heap
- Win32::Win32Interlocked
  - Threading::Interlocked
- Win32::Win32IpAddress
- Win32::Win32MiniDump
  - Debug::MiniDump
- Win32::Win32Timer
  - Timing::Timer
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- a -

- abs() : Math::float2
- AbstractLightEntity() : Lighting::AbstractLightEntity
- Accept() : Win32::Win32Socket
- AcceptsMessage() : Messaging::Port, Game::Entity
- AcceptsRequest() : Debug::IoPageHandler, Debug::MemoryPageHandler, Debug::ScriptingPageHandler, Debug::DisplayPageHandler, Debug::CorePageHandler, Debug::MeshPageHandler, Debug::ShaderPageHandler, Http::HttpRequestHandler, Debug::TexturePageHandler
- AccessMode : IO::Stream
- AccessPattern : IO::Stream
- AccessType : CoreGraphics::VertexComponent
- ActivateEntity() : BaseGameFeature::EntityManager
- ActorEntity() : Graphics::ActorEntity
- AdapterExists() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice
- AdapterInfo() : CoreGraphics::AdapterInfo
- Add() : Win32::Win32Interlocked, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Models::VisResolveContainer< TYPE >
- AddAfter() : Util::List< TYPE >
- AddArg() : Scripting::ArgsBlock
- AddAttr() : Attr::AttributeContainer, Http::HtmlPageWriter
- AddBack() : Util::List< TYPE >
- AddBatch() : Frame::FramePass
- AddBefore() : Util::List< TYPE >
- AddBlob() : Game::Entity
- AddBool() : Game::Entity
- AddCategoryAttr() : BaseGameFeature::CategoryManager
- AddChild() : Models::ModelNode, Models::ModelNodeInstance
- AddColorBuffer() : Base::RenderTargetBase
- AddColumn() : Attr::AttributeTable
- AddDelayedJob() : BaseGameFeature::EntityManager
- AddDependency() : Graphics::View
- AddDepthStencilBuffer() : Base::RenderTargetBase
- AddDirEntry() : IO::ZipDirEntry
- AddFileEntry() : IO::ZipDirEntry
- AddFloat() : Game::Entity
- AddFloat4() : Game::Entity
- AddFramePass() : Frame::FrameShader
- AddFront() : Util::List< TYPE >
- AddGuid() : Game::Entity
- AddInt() : Game::Entity
- AddLink() : Graphics::GraphicsEntity
- AddList() : Util::List< TYPE >
- AddMatrix44() : Game::Entity
- AddPostEffect() : Frame::FrameShader
- AddPreShader() : Base::ShaderInstanceBase
- AddProperties() : BaseGameFeature::FactoryManager
- AddRef() : Core::RefCounted, Util::SimpleTree< VALUETYPE >
- AddRenderTarget() : Frame::FrameShader
- AddRow() : Attr::AttributeTable
- AddStateHandler() : App::GameApplication
- AddString() : Game::Entity
- AddVariable() : Frame::FrameBatch, Frame::FramePass, Frame::FramePostEffect
- AddVisibleNodeInstance() : Models::ModelNode
- AdvanceProgress() : BaseGameFeature::LoaderServer
- all() : Math::float2
- Alloc() : Win32::Win32Heap
- ANS1toUTF8() : Util::String
- any() : Math::float2
- Append() : Util::Array< TYPE >, Util::SimpleTree<
VALUETYPE >::Node, Util::String
- AppendArray() : Util::Array< TYPE >
- AppendBool() : Util::String
- AppendFloat() : Util::String
- AppendFloat4() : Util::String
- AppendInt() : Util::String
- AppendLocalPath() : IO::URI
- AppendMatrix44() : Util::String
- AppendRange() : Util::String
- Application() : App::Application
- Apply() : Base::ShaderVariableInstanceBase
- ApplyImpulseAtPos() : PhysicsFeature::PhysicsProperty
- ApplyModelEntityLights() : Lighting::LightServerBase, Lighting::SM30LightServer
- ApplyModelTransforms() : Base::TransformDeviceBase
- ApplyPrimitives() : Base::MeshBase
- ApplySharedState() : Models::ModelNode, Models::ShapeNode, Models::SkinShapeNode, Models::StateNode
- ApplyState() : Models::ModelNodeInstance, Models::StateNodeInstance, Models::TransformNodeInstance
- ApplyViewSettings() : Base::TransformDeviceBase
- AreMipMapsEnabled() : Base::RenderTargetBase
- ArgsBlock() : Scripting::ArgsBlock
- Arguments() : Scripting::Command
- ArgValue() : Scripting::ArgsBlock
- Array() : Util::Array< TYPE >
- AsBinary() : Win32::Win32Guid
- AsBool() : Util::String
- AsCharPtr() : Util::String
- AsD3D9MultiSampleType() : Direct3D9::D3D9Types
- AsD3D9PixelFormat() : Direct3D9::D3D9Types
- AsD3D9PrimitiveType() : Direct3D9::D3D9Types
- AsD3D9VertexDeclarationType() : Direct3D9::D3D9Types
- AsD3D9VertexDeclarationUsage() : Direct3D9::D3D9Types
- AsD3DXImageFileFormat() : Direct3D9::D3D9Types
- AsFloat() : Util::String
- AsFloat4() : Util::String
- AsInt() : Util::String
- AsMatrix44() : Util::String
- AsNebulaPixelFormat() : Direct3D9::D3D9Types
- Assign() : IO::Assign
- AsString() : Win32::Win32Guid, IO::MediaType, IO::URI, Util::FourCC
- AsUInt() : Util::FourCC
- AsyncGraphicsHandler() : AsyncGraphics::AsyncGraphicsHandler
- AsyncGraphicsInterface() : AsyncGraphics::AsyncGraphicsInterface
- AsyncHttpInterface() : AsyncHttp::AsyncHttpInterface
- AsyncPort() : Messaging::AsyncPort
- AsyncRenderApplication() : App::AsyncRenderApplication
- AsyncViewerApplication() : App::AsyncViewerApplication
- At() : Util::FixedTable< TYPE >, Util::Array< TYPE >
- Atom() : Util::Atom< TYPE >
- AttachChildCell() : Graphics::Cell
- AttachDisplayEventHandler() : AsyncGraphics::DisplayProxy
- AttachEntity() : BaseGameFeature::EntityManager, Graphics::Stage, Graphics::Cell
- AttachEntityLoader() : BaseGameFeature::LoaderServer
- AttachEventHandler() : Base::DisplayDeviceBase, Base::RenderDeviceBase
- AttachGameFeature() : Game::GameServer
- AttachHandler() : Messaging::AsyncPort, Messaging::Port, IO::Console
- AttachInputHandler() : Base::InputServerBase
- AttachManager() : Game::FeatureUnit
- AttachMapper() : Resources::ResourceManager
- AttachNode() : Models::Model
- AttachPort() : Messaging::Dispatcher
- AttachRenderEventHandler() : AsyncGraphics::DisplayProxy
- AttachRequestHandler() : Http::HttpServer
- AttachVisibleLight() : Lighting::LightServerBase
- AttachVisibleModelInstance() : Models::VisResolver
- Attribute() : Attr::Attribute
- AttributeContainer() : Attr::AttributeContainer
- AttributeDefinition() : Attr::AttributeDefinition< VALUETYPE,
TYPE >
- AttributeDefinitionBase() : Attr::AttributeDefinitionBase
- AttributeTable() : Attr::AttributeTable
- AttrId() : Attr::AttrId

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
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:

- App
- Application
- AsyncGraphics
- Attr
- Base
- BaseGameFeature
- Commands
- Core
- CoreFeature
- CoreGraphics
- Debug
- Direct3D9
- Frame
- Game
- Graphics
- GraphicsFeature
- Http
- Input
- Interface
- IO
- Legacy
- Lighting
- Loader
- Math
- Memory
- Messaging
- Models
- Net
- PhysicsFeature
- RenderUtil
- Resources
- Scripting
- System
- Threading
The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
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
//--- nviewer3.cc
// (C) 2007 Radon Labs GmbH
#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>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>ConsoleApplication</td>
</tr>
<tr>
<td>AsyncRenderApplication</td>
</tr>
<tr>
<td>AsyncViewerApplication</td>
</tr>
<tr>
<td>RenderApplication</td>
</tr>
<tr>
<td>ViewerApplication</td>
</tr>
<tr>
<td>GameApplication</td>
</tr>
<tr>
<td>GameStateHandler</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Tue Feb 19 12:16:40 2008
- 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

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).

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:41 2008
AsyncGraphics Namespace Reference
Detailed Description

FIXME!

This file was generated with Nebula3's idlc compiler tool. DO NOT EDIT

class AsyncGraphics::StageProxy

A client-side proxy of a Graphics::Stage in the AsyncGraphics subsystem. The StageProxy offers a friendly frontend to the client thread, and communicates with its server-side Stage object through the AsyncGraphicsInterface method. There is a 1:1 relationship between the server-side Stage and the client-side StageProxy, thus the StageProxy may safely store read-only information on the client side without synchronizing with its Stage.

(C) 2007 Radon Labs GmbH
## Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsyncGraphicsHandler</td>
<td></td>
</tr>
<tr>
<td>AsyncGraphicsInterface</td>
<td></td>
</tr>
<tr>
<td>CameraEntityProxy</td>
<td></td>
</tr>
<tr>
<td>DisplayProxy</td>
<td></td>
</tr>
<tr>
<td>GraphicsEntityProxy</td>
<td></td>
</tr>
<tr>
<td>GraphicsServerProxy</td>
<td></td>
</tr>
<tr>
<td>ViewProxy</td>
<td></td>
</tr>
<tr>
<td>Handle</td>
<td></td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
Attr Namespace Reference
Detailed Description

FIXME!

This is the central attribute registry for this feature. For more information on attributes, see Attr::Attribute.

(C) 2007 Radon Labs GmbH

A specialized graphics property for actors. This creates a Graphics::ActorEntity, knows how to switch animations and manages attachments.

(C) 2005 Radon Labs GmbH

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

A chase camera for 3rd person camera control.

(C) 2005 Radon Labs GmbH

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

A light property adds a light source object (Graphics::LightEntity) to a game entity.

(C) 2005 Radon Labs GmbH

(C) 2005 Radon Labs GmbH

This is the central attribute registry for this feature. For more information on attributes, see Attr::Attribute.

(C) 2007 Radon Labs GmbH

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) 2005 Radon Labs GmbH
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Attribute</code></td>
</tr>
<tr>
<td><code>AttributeContainer</code></td>
</tr>
<tr>
<td><code>AttributeDefinition</code></td>
</tr>
<tr>
<td><code>AttributeDefinitionBase</code></td>
</tr>
<tr>
<td><code>AttributeTable</code></td>
</tr>
<tr>
<td><code>AttrId</code></td>
</tr>
<tr>
<td><code>BlobAttrId</code></td>
</tr>
<tr>
<td><code>BoolAttrId</code></td>
</tr>
<tr>
<td><code>Float4AttrId</code></td>
</tr>
<tr>
<td><code>FloatAttrId</code></td>
</tr>
<tr>
<td><code>GuidAttrId</code></td>
</tr>
<tr>
<td><code>IntAttrId</code></td>
</tr>
<tr>
<td><code>Matrix44AttrId</code></td>
</tr>
<tr>
<td><code>StringAttrId</code></td>
</tr>
<tr>
<td><code>AccessMode</code></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Tue Feb 19 12:16:41 2008
Base Namespace Reference
Detailed Description

The Nebula3 Base Namespace

The **Base** subsystem contains all base classes where platform-specific 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 functionality of all RenderDevice classes.

The main reason why the **Base** namespace has been introduced is to not contaminate the autodocs and IntelliSense with unrelated information.
# Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CalendarTimeBase</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>RenderDeviceBase</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>TextureBase</code></td>
</tr>
<tr>
<td><code>TransformDeviceBase</code></td>
</tr>
<tr>
<td><code>VertexBufferBase</code></td>
</tr>
<tr>
<td><code>VertexLayoutBase</code></td>
</tr>
<tr>
<td><code>GamePadBase</code></td>
</tr>
<tr>
<td><code>InputServerBase</code></td>
</tr>
<tr>
<td><code>KeyboardBase</code></td>
</tr>
<tr>
<td><code>MouseBase</code></td>
</tr>
<tr>
<td><code>D3D9TransformDevice</code></td>
</tr>
<tr>
<td><code>PreShader</code></td>
</tr>
</tbody>
</table>
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 enviroment entities.

The time property adds the attribute "Time" to the entity. This attributes contains the time since the time property has been attached to the entity.

(C) 2007 Radon Labs GmbH

Entities with this property can be transformed.

(C) 2007 Radon Labs GmbH
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntityLoader</td>
<td></td>
</tr>
<tr>
<td>EntityLoaderBase</td>
<td></td>
</tr>
<tr>
<td>EnvironmentLoader</td>
<td></td>
</tr>
<tr>
<td>LevelLoader</td>
<td></td>
</tr>
<tr>
<td>LoaderServer</td>
<td></td>
</tr>
<tr>
<td>UserProfile</td>
<td></td>
</tr>
<tr>
<td>CategoryManager</td>
<td></td>
</tr>
<tr>
<td>EntityManager</td>
<td></td>
</tr>
<tr>
<td>EnvEntityManager</td>
<td></td>
</tr>
<tr>
<td>EnvQueryManager</td>
<td></td>
</tr>
<tr>
<td>FactoryManager</td>
<td></td>
</tr>
<tr>
<td>FocusManager</td>
<td></td>
</tr>
<tr>
<td>GlobalAttrsManager</td>
<td></td>
</tr>
<tr>
<td>MoveFollow</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:43 2008
Commands Namespace Reference
Detailed Description

This file was generated with Nebula3's idlc compiler tool. DO NOT EDIT

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:43 2008
- 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 lifetime 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 \texttt{Core::RefCounted}.

Deriving from the \texttt{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 \texttt{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 \texttt{DeclareClass()} macro, the default constructor and the virtual destructor and the \texttt{RegisterClass()} macro outside of the class declaration. The \texttt{DeclareClass()} macro adds some minimal Nebula3-specific information to the class declaration for the RTTI and factory mechanism. The \texttt{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
neglige, 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 `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 exisstance 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;
}
```
/**
 * Implements the Singleton destructor.
 */
MySingletonClass::~MySingletonClass()
{
    DestructSingleton;
}

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 singe threaded 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>CoreServer</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>Factory</td>
</tr>
<tr>
<td>class</td>
<td>RefCounted</td>
</tr>
<tr>
<td>class</td>
<td>RefCountedList</td>
</tr>
<tr>
<td>class</td>
<td>Rtti</td>
</tr>
<tr>
<td>class</td>
<td>SysFunc</td>
</tr>
<tr>
<td>class</td>
<td>Ptr</td>
</tr>
<tr>
<td>class</td>
<td>Singleton</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by `doxygen` at Tue Feb 19 12:16:43 2008
CoreFeature Namespace Reference
Detailed Description

The CoreFeature creates the fundamental classes of Nebula3 used for a new game application. It creates Core, IO, Http (for debugging) and script servers.
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
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>CPUIndexBuffer</td>
</tr>
<tr>
<td>CPUMemoryIndexBufferLoader</td>
</tr>
<tr>
<td>CPUMemoryVertexBufferLoader</td>
</tr>
<tr>
<td>CPUVertexBuffer</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>MemoryVertexBufferLoader</td>
</tr>
<tr>
<td>Mesh</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>RenderTarget</td>
</tr>
<tr>
<td>Shader</td>
</tr>
<tr>
<td>ShaderFeature</td>
</tr>
<tr>
<td>ShaderInstance</td>
</tr>
<tr>
<td>ShaderServer</td>
</tr>
<tr>
<td>Class</td>
</tr>
<tr>
<td>-----------------------</td>
</tr>
<tr>
<td>ShaderVariable</td>
</tr>
<tr>
<td>ShaderVariableInstance</td>
</tr>
<tr>
<td>ShaderVariation</td>
</tr>
<tr>
<td>ShapeRenderer</td>
</tr>
<tr>
<td>StreamAnimationLoader</td>
</tr>
<tr>
<td>StreamMeshLoader</td>
</tr>
<tr>
<td>StreamShaderLoader</td>
</tr>
<tr>
<td>StreamTextureLoader</td>
</tr>
<tr>
<td>StreamTextureSaver</td>
</tr>
<tr>
<td>Texture</td>
</tr>
<tr>
<td>ThreadSafeDisplayEventExceptionHandler</td>
</tr>
<tr>
<td>ThreadSafeRenderEventExceptionHandler</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>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:44 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
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>MiniDump</td>
</tr>
<tr>
<td>IoPageHandler</td>
</tr>
<tr>
<td>MemoryPageHandler</td>
</tr>
<tr>
<td>ScriptingPageHandler</td>
</tr>
<tr>
<td>DisplayPageHandler</td>
</tr>
<tr>
<td>MeshPageHandler</td>
</tr>
<tr>
<td>ShaderPageHandler</td>
</tr>
<tr>
<td>TexturePageHandler</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:45 2008
Direct3D9 Namespace Reference
Detailed Description

FIXME!
### Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3D9DisplayDevice</td>
</tr>
<tr>
<td>D3D9IndexBuffer</td>
</tr>
<tr>
<td>D3D9MemoryIndexBufferLoader</td>
</tr>
<tr>
<td>D3D9MemoryVertexBufferLoader</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>D3D9ShapeRenderer</td>
</tr>
<tr>
<td>D3D9StreamShaderLoader</td>
</tr>
<tr>
<td>D3D9StreamTextureLoader</td>
</tr>
<tr>
<td>D3D9StreamTextureSaver</td>
</tr>
<tr>
<td>D3D9Texture</td>
</tr>
<tr>
<td>D3D9Types</td>
</tr>
<tr>
<td>D3D9VertexBuffer</td>
</tr>
<tr>
<td>D3D9VertexLayout</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](http://www.doxygen.org) at Tue Feb 19 12:16:45 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
Frame Namespace Reference
Detailed Description

FIXME!
## Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>FrameBatch</td>
</tr>
<tr>
<td>class</td>
<td>FramePass</td>
</tr>
<tr>
<td>class</td>
<td>FramePostEffect</td>
</tr>
<tr>
<td>class</td>
<td>FrameServer</td>
</tr>
<tr>
<td>class</td>
<td>FrameShader</td>
</tr>
<tr>
<td>class</td>
<td>FrameShaderLoader</td>
</tr>
<tr>
<td>class</td>
<td>LightingMode</td>
</tr>
<tr>
<td>class</td>
<td>SortingMode</td>
</tr>
</tbody>
</table>
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- Namespace List
- Namespace Members
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>
<th>Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>FeatureUnit</td>
</tr>
<tr>
<td>class</td>
<td>GameServer</td>
</tr>
<tr>
<td>class</td>
<td>Manager</td>
</tr>
<tr>
<td>class</td>
<td>Property</td>
</tr>
<tr>
<td>class</td>
<td>BaseGameFeatureUnit</td>
</tr>
<tr>
<td>class</td>
<td>CoreFeature</td>
</tr>
<tr>
<td>class</td>
<td>PhysicsFeatureUnit</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by `doxygen` at Tue Feb 19 12:16:46 2008
Graphics Namespace Reference
Detailed Description

FIXME!

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
## Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActorEntity</td>
</tr>
<tr>
<td>CameraEntity</td>
</tr>
<tr>
<td>Cell</td>
</tr>
<tr>
<td>GraphicsEntity</td>
</tr>
<tr>
<td>GraphicsServer</td>
</tr>
<tr>
<td>ModelEntity</td>
</tr>
<tr>
<td>QuadtreeStageBuilder</td>
</tr>
<tr>
<td>SimpleStageBuilder</td>
</tr>
<tr>
<td>Stage</td>
</tr>
<tr>
<td>StageBuilder</td>
</tr>
<tr>
<td>View</td>
</tr>
</tbody>
</table>
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.

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

An basic point'n'click third person input property which creates Goto messages when clicking into the environment.

(C) 2005 Radon Labs GmbH
Data Structures

- class CameraDistance
- class CameraFocus
- class CameraOrbit
- class CameraReset
- class GetGraphicsEntities
- class InputFocus
- class GraphicsFeatureUnitUnit
- class SetVisible
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>
</tr>
</thead>
<tbody>
<tr>
<td>Base64Writer</td>
</tr>
<tr>
<td>DefaultHttpRequestHandler</td>
</tr>
<tr>
<td>HtmlElement</td>
</tr>
<tr>
<td>HtmlPageWriter</td>
</tr>
<tr>
<td>HttpMethod</td>
</tr>
<tr>
<td>HttpRequest</td>
</tr>
<tr>
<td>HttpRequestHandler</td>
</tr>
<tr>
<td>HttpRequestReader</td>
</tr>
<tr>
<td>HttpResponseWriter</td>
</tr>
<tr>
<td>HttpServer</td>
</tr>
<tr>
<td>HttpStatus</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:47 2008
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 Tue Feb 19 12:16:47 2008
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

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyFile</td>
</tr>
<tr>
<td>CreateDirectory</td>
</tr>
<tr>
<td>DeleteDirectory</td>
</tr>
<tr>
<td>DeleteFile</td>
</tr>
<tr>
<td>IOMessage</td>
</tr>
<tr>
<td>MountZipArchive</td>
</tr>
<tr>
<td>ReadStream</td>
</tr>
<tr>
<td>WriteStream</td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Tue Feb 19 12:16:48 2008
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:

```
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 invocations.

- **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
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:

```
file:///c:/temp/bla.txt
file://samba/temp/bla.txt
http://www.radonlabs.de/index.html
http://user:password@www.myserver.com:8080/index.html#main
```

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::TextWriter/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
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>
</tr>
</thead>
<tbody>
<tr>
<td>Assign</td>
</tr>
<tr>
<td>BinaryReader</td>
</tr>
<tr>
<td>BinaryWriter</td>
</tr>
<tr>
<td>Console</td>
</tr>
<tr>
<td>ConsoleHandler</td>
</tr>
<tr>
<td>FileStream</td>
</tr>
<tr>
<td>FileTime</td>
</tr>
<tr>
<td>IoServer</td>
</tr>
<tr>
<td>MediaType</td>
</tr>
<tr>
<td>MemoryStream</td>
</tr>
<tr>
<td>Stream</td>
</tr>
<tr>
<td>StreamReader</td>
</tr>
<tr>
<td>StreamWriter</td>
</tr>
<tr>
<td>TextReader</td>
</tr>
<tr>
<td>TextWriter</td>
</tr>
<tr>
<td>URI</td>
</tr>
<tr>
<td>XmlReader</td>
</tr>
<tr>
<td>XmlWriter</td>
</tr>
<tr>
<td>ZipFileStream</td>
</tr>
<tr>
<td>ZipArchive</td>
</tr>
<tr>
<td>ZipDirEntry</td>
</tr>
<tr>
<td>ZipFileEntry</td>
</tr>
<tr>
<td>ZipFileSystem</td>
</tr>
<tr>
<td>IOInterfaceHandlerBase</td>
</tr>
<tr>
<td>Interface</td>
</tr>
</tbody>
</table>
Main Page  
Namespaces  
Data Structures  
Files  
Related Pages  

Namespace List  
Namespace Members
Legacy Namespace Reference
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 Nax2StreamReader
class Nvx2StreamReader
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

FIXME!
## Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractLightEntity</td>
<td></td>
</tr>
<tr>
<td>LightServerBase</td>
<td></td>
</tr>
<tr>
<td>GlobalLightEntity</td>
<td></td>
</tr>
<tr>
<td>LightServer</td>
<td></td>
</tr>
<tr>
<td>LightType</td>
<td></td>
</tr>
<tr>
<td>ShadowServer</td>
<td></td>
</tr>
<tr>
<td>PSSMUtil</td>
<td></td>
</tr>
<tr>
<td>SM30LightServer</td>
<td></td>
</tr>
<tr>
<td>SM30ShadowServer</td>
<td></td>
</tr>
<tr>
<td>SpotLightEntity</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by [doxygen](https://www.doxygen.org/) at Tue Feb 19 12:16:49 2008
- 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.
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>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>bbox</td>
</tr>
<tr>
<td>class</td>
<td>ClipStatus</td>
</tr>
<tr>
<td>class</td>
<td>point</td>
</tr>
<tr>
<td>class</td>
<td>vector</td>
</tr>
<tr>
<td>class</td>
<td>float2</td>
</tr>
<tr>
<td>class</td>
<td>line</td>
</tr>
<tr>
<td>class</td>
<td>noise</td>
</tr>
<tr>
<td>class</td>
<td>polar</td>
</tr>
<tr>
<td>class</td>
<td>rectangle</td>
</tr>
<tr>
<td>class</td>
<td>sphere</td>
</tr>
<tr>
<td>class</td>
<td>transform44</td>
</tr>
<tr>
<td>class</td>
<td>float4</td>
</tr>
<tr>
<td>class</td>
<td>matrix44</td>
</tr>
<tr>
<td>class</td>
<td>plane</td>
</tr>
<tr>
<td>class</td>
<td>quaternion</td>
</tr>
</tbody>
</table>
## Functions

<table>
<thead>
<tr>
<th>__forceinline scalar</th>
<th><strong>n_log2</strong> (scalar f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>__forceinline int</td>
<td><strong>n_iclamp</strong> (int val, int minVal, int maxVal)</td>
</tr>
<tr>
<td>__forceinline bool</td>
<td><strong>n_fequal</strong> (scalar f0, scalar f1, scalar tol)</td>
</tr>
<tr>
<td>__forceinline bool</td>
<td><strong>n_fless</strong> (scalar f0, scalar f1, scalar tol)</td>
</tr>
<tr>
<td>__forceinline bool</td>
<td><strong>n_fgreater</strong> (scalar f0, scalar f1, scalar tol)</td>
</tr>
<tr>
<td>__forceinline scalar</td>
<td><strong>n_smooth</strong> (scalar newVal, scalar curVal, scalar maxChange)</td>
</tr>
<tr>
<td>__forceinline scalar</td>
<td><strong>n_clamp</strong> (scalar val, scalar lower, scalar upper)</td>
</tr>
<tr>
<td>__forceinline scalar</td>
<td><strong>n_saturate</strong> (scalar val)</td>
</tr>
<tr>
<td>__forceinline scalar</td>
<td><strong>n_rand</strong> ()</td>
</tr>
<tr>
<td>__forceinline scalar</td>
<td><strong>n_rand</strong> (scalar min, scalar max)</td>
</tr>
<tr>
<td>__forceinline int</td>
<td><strong>n_fchop</strong> (scalar f)</td>
</tr>
<tr>
<td>__forceinline int</td>
<td><strong>n_frnd</strong> (scalar f)</td>
</tr>
<tr>
<td>__forceinline float</td>
<td><strong>n_lerp</strong> (scalar x, scalar y, scalar l)</td>
</tr>
<tr>
<td>__forceinline scalar</td>
<td><strong>n_modangle</strong> (scalar a)</td>
</tr>
<tr>
<td>__forceinline scalar</td>
<td><strong>n_angulardistance</strong> (scalar from, scalar to)</td>
</tr>
</tbody>
</table>
Function Documentation

__forceinline
scalar ( scalar f )
Math::n_log2

log2() function.

__forceinline
int ( int val,
Math::n_iclamp
    int minVal,
    int maxVal
)

Integer clamping.

__forceinline
bool ( scalar f0,
Math::n_fequal
    scalar f1,
    scalar tol
)

A fuzzy floating point equality check

__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-then check.

__forceinline__
scalar Math::n_smooth(scalar newVal, scalar curVal, scalar maxChange)

Smooth a new value towards an old value using a change value.

__forceinline__
scalar Math::n_clamp(scalar val, scalar lower, scalar upper)

Clamp a value against lower and upper boundary.

__forceinline__
scalar Math::n_saturate(scalar val)

Saturate a value (clamps between 0.0f and 1.0f)

__forceinline__
scalar Math::n_rand()

Return a pseudo random number between 0 and 1.

__forceinline__
scalar Math::n_rand(scalar min, scalar max)

Return a pseudo random number between min and max.

__forceinline__
int Math::n_fchop(scalar f)

Return an integer from a fuzzy floating point number.
Chop float to int.

```cpp
__forceinline
int Math::n_frnd(scalar f)
```

Round float to integer.

```cpp
__forceinline
float Math::n_lerp(scalar x, scalar y, scalar l)
```

Linearly interpolate between 2 values: 
$$\text{ret} = x + l \times (y - x)$$

```cpp
__forceinline
scalar Math::n_modangle(scalar a)
```

Normalize an angular value into the range rad(0) to rad(360).

```cpp
__forceinline scalar Math::n_angulardistance(scalar from, scalar to)
```

Get angular distance.
Memory Namespace Reference
Detailed Description

The Nebula3 Memory subsystem implements custom memory allocation mechanisms which provide higher performance and better debugging aids.
<table>
<thead>
<tr>
<th></th>
<th>MemoryStatus</th>
<th>Heap</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Alloc</code></td>
<td>void *(size_t size)</td>
</tr>
<tr>
<td><code>Realloc</code></td>
<td>void *(void *ptr, size_t size)</td>
</tr>
<tr>
<td><code>Free</code></td>
<td>void *(void *ptr)</td>
</tr>
<tr>
<td><code>Copy</code></td>
<td>void *(const void *from, void *to, size_t numBytes)</td>
</tr>
<tr>
<td><code>Clear</code></td>
<td>void *(void *ptr, size_t numBytes)</td>
</tr>
<tr>
<td><code>DuplicateCString</code></td>
<td>char *(const char *from)</td>
</tr>
</tbody>
</table>
Function Documentation

__forceinline
void* ( size_t size )
Memory::Alloc

Allocate a block of memory from the process heap.

__forceinline
void* ( void * ptr,
Memory::Realloc
    size_t size
)

Reallocate a block of memory.

__forceinline
void ( *ptr )
Memory::Free

Free a chunk of memory from the process heap.

__forceinline
void ( const char * from,
void * to,
size_t numBytes )
Memory::Copy

Copy a chunk of memory (note the argument order is different from memcpy()!!)

__forceinline
void ( void * ptr,
    size_t numBytes
)  
Memory::Clear

Overwrite a chunk of memory with 0's.

__forceinline char* ( char *from )
Memory::DuplicateCString  *

Duplicate a 0-terminated string.
Messaging Namespace Reference
Detailed Description

FIXME!
### Data Structures

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsyncPort</code></td>
<td></td>
</tr>
<tr>
<td><code>Dispatcher</code></td>
<td></td>
</tr>
<tr>
<td><code>Handler</code></td>
<td></td>
</tr>
<tr>
<td><code>Id</code></td>
<td></td>
</tr>
<tr>
<td><code>Message</code></td>
<td></td>
</tr>
<tr>
<td><code>MessageReader</code></td>
<td></td>
</tr>
<tr>
<td><code>MessageWriter</code></td>
<td></td>
</tr>
<tr>
<td><code>Port</code></td>
<td></td>
</tr>
<tr>
<td><code>MoveDirection</code></td>
<td></td>
</tr>
<tr>
<td><code>MoveGoto</code></td>
<td></td>
</tr>
<tr>
<td><code>MoveRotate</code></td>
<td></td>
</tr>
<tr>
<td><code>MoveSetVelocity</code></td>
<td></td>
</tr>
<tr>
<td><code>MoveStop</code></td>
<td></td>
</tr>
<tr>
<td><code>MoveTurn</code></td>
<td></td>
</tr>
<tr>
<td><code>SetTransform</code></td>
<td></td>
</tr>
<tr>
<td><code>UpdateTransform</code></td>
<td></td>
</tr>
<tr>
<td><code>AddAttachment</code></td>
<td></td>
</tr>
<tr>
<td><code>AddSkin</code></td>
<td></td>
</tr>
<tr>
<td><code>AnimationHotspotTriggered</code></td>
<td></td>
</tr>
<tr>
<td><code>FadeAnimation</code></td>
<td></td>
</tr>
<tr>
<td><code>GetActiveAnimation</code></td>
<td></td>
</tr>
<tr>
<td><code>GetAnimationInfo</code></td>
<td></td>
</tr>
<tr>
<td><code>GetAttachmentEntities</code></td>
<td></td>
</tr>
<tr>
<td><code>GetHotspotTime</code></td>
<td></td>
</tr>
<tr>
<td><code>GetJointMatrix</code></td>
<td></td>
</tr>
<tr>
<td><code>HasAttachment</code></td>
<td></td>
</tr>
<tr>
<td><code>HideAttachment</code></td>
<td></td>
</tr>
<tr>
<td><code>RemAttachment</code></td>
<td></td>
</tr>
<tr>
<td><code>RemSkin</code></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>SetAnimation</td>
<td></td>
</tr>
<tr>
<td>SetFadeAnimationMix</td>
<td></td>
</tr>
<tr>
<td>ShowAttachment</td>
<td></td>
</tr>
<tr>
<td>UpdateAttachments</td>
<td></td>
</tr>
<tr>
<td>ApplyImpulseAtPos</td>
<td></td>
</tr>
<tr>
<td>GetPhysicsEntity</td>
<td></td>
</tr>
</tbody>
</table>
Models Namespace Reference
Detailed Description

FIXME!
# Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinaryModelReader</td>
</tr>
<tr>
<td>ModelReader</td>
</tr>
<tr>
<td>N2ModelReader</td>
</tr>
<tr>
<td>StreamModelLoader</td>
</tr>
<tr>
<td>XmlModelReader</td>
</tr>
<tr>
<td>ManagedModel</td>
</tr>
<tr>
<td>Model</td>
</tr>
<tr>
<td>ModelInstance</td>
</tr>
<tr>
<td>ModelNode</td>
</tr>
<tr>
<td>ModelNodeInstance</td>
</tr>
<tr>
<td>ModelNodeType</td>
</tr>
<tr>
<td>ModelServer</td>
</tr>
<tr>
<td>CharacterNode</td>
</tr>
<tr>
<td>CharacterNodeInstance</td>
</tr>
<tr>
<td>ParticleSystemNode</td>
</tr>
<tr>
<td>ParticleSystemNodeInstance</td>
</tr>
<tr>
<td>ShapeNode</td>
</tr>
<tr>
<td>ShapeNodeInstance</td>
</tr>
<tr>
<td>SkinShapeNode</td>
</tr>
<tr>
<td>SkinShapeNodeInstance</td>
</tr>
<tr>
<td>StateNode</td>
</tr>
<tr>
<td>StateNodeInstance</td>
</tr>
<tr>
<td>TransformNode</td>
</tr>
<tr>
<td>TransformNodeInstance</td>
</tr>
<tr>
<td>BinaryModelWriter</td>
</tr>
<tr>
<td>ModelWriter</td>
</tr>
<tr>
<td>XmlModelWriter</td>
</tr>
<tr>
<td>VisResolveContainer</td>
</tr>
<tr>
<td>VisResolver</td>
</tr>
</tbody>
</table>
Net Namespace Reference
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:

```java
// 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:

```java
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>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Socket</strong></td>
<td></td>
</tr>
<tr>
<td><strong>TcpClient</strong></td>
<td></td>
</tr>
<tr>
<td><strong>TcpClientConnection</strong></td>
<td></td>
</tr>
<tr>
<td><strong>TcpServer</strong></td>
<td></td>
</tr>
<tr>
<td><strong>IpAddress</strong></td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by **doxygen** at Tue Feb 19 12:16:52 2008
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.

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) 2005 Radon Labs GmbH
Data Structures

class MouseGripperProperty

class PhysicsProperty
RenderUtil Namespace Reference
Detailed Description

FIXME!
class MayaCameraUtil
Detailed Description

FIXME!
### Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynamicMeshResourceLoader</td>
</tr>
<tr>
<td>ManagedMesh</td>
</tr>
<tr>
<td>ManagedResource</td>
</tr>
<tr>
<td>ManagedTexture</td>
</tr>
<tr>
<td>Resource</td>
</tr>
<tr>
<td>ResourceLoader</td>
</tr>
<tr>
<td>ResourceManager</td>
</tr>
<tr>
<td>ResourceMapper</td>
</tr>
<tr>
<td>ResourceSaver</td>
</tr>
<tr>
<td>SharedResourceServer</td>
</tr>
<tr>
<td>SimpleResourceMapper</td>
</tr>
<tr>
<td>ResourceId</td>
</tr>
</tbody>
</table>
• Main Page
• Namespaces
• Data Structures
• Files
• Related Pages

• Namespace List
• Namespace Members
Scripting Namespace Reference
Detailed Description

FIXME!
Data Structures

- class ArgsBlock
- class Command
- class LuaServer
- class ScriptServer
- class Arg
System Namespace Reference
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

<table>
<thead>
<tr>
<th>class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ByteOrder</td>
<td></td>
</tr>
<tr>
<td>Cpu</td>
<td></td>
</tr>
<tr>
<td>Win32Registry</td>
<td></td>
</tr>
<tr>
<td>AppEntry</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:53 2008
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 SafePriorityQueue
class SafeQueue
class Thread
class ThreadLocalPtr
class Barrier
Timing Namespace Reference
Detailed Description

The Nebula3 **Timing** subsystem offers classes for measuring absolute and elapsed time, or obtaining the current date and wall clock time.
## Data Structures

<table>
<thead>
<tr>
<th>class</th>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>CalendarTime</td>
</tr>
<tr>
<td>class</td>
<td>Timer</td>
</tr>
<tr>
<td>class</td>
<td>TimingTimeSource</td>
</tr>
<tr>
<td>class</td>
<td>InputTimeSource</td>
</tr>
<tr>
<td>class</td>
<td>SystemTimeSource</td>
</tr>
<tr>
<td>class</td>
<td>TimeManager</td>
</tr>
<tr>
<td>class</td>
<td>TimeSource</td>
</tr>
</tbody>
</table>
typedef double Time

the time datatype
Functions

```c
void Sleep (Time t)
```
Function Documentation

```c
void Timing::Sleep (Time t) [inline]
```

Put current thread to sleep for specified amount of seconds.
Util Namespace Reference
Detailed Description

The Nebula3 Util Subsystem

The **Util** namespace is a toolbox of general utility classes, mainly containers and a powerful string class.
# Data Structures

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
</tr>
<tr>
<td>Atom</td>
</tr>
<tr>
<td>Blob</td>
</tr>
<tr>
<td>CmdLineArgs</td>
</tr>
<tr>
<td>Crc</td>
</tr>
<tr>
<td>Dictionary</td>
</tr>
<tr>
<td>FixedArray</td>
</tr>
<tr>
<td>FixedTable</td>
</tr>
<tr>
<td>FourCC</td>
</tr>
<tr>
<td>HashTable</td>
</tr>
<tr>
<td>KeyValuePair</td>
</tr>
<tr>
<td>List</td>
</tr>
<tr>
<td>Proxy</td>
</tr>
<tr>
<td>Queue</td>
</tr>
<tr>
<td>SimpleTree</td>
</tr>
<tr>
<td>Stack</td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td>Variant</td>
</tr>
<tr>
<td>Guid</td>
</tr>
<tr>
<td>StringAtom</td>
</tr>
<tr>
<td>CharEnhancementUtil</td>
</tr>
<tr>
<td>LightFlickerUtil</td>
</tr>
<tr>
<td>SegmentedGfxUtil</td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Namespace List
Namespace Members
Win32 Namespace Reference
Detailed Description

[TODO: Describe Win32 subsystem]
Data Structures

class SysFunc
class Win32MiniDump
class Win32ConsoleHandler
class Win32FileTime
class Win32FSWrapper
class Win32Heap
class Win32IpAddress
class Win32Socket
class Win32CriticalSection
class Win32Event
class Win32Interlocked
class Win32Thread
class Win32CalendarTime
class Win32Timer
class Win32Guid
class Win32DisplayDevice
class Win32InputDisplayEventHandler
class Win32InputServer
class Win32Mouse
class Win32Barrier
Here is a list of all documented namespace members with links to the namespaces they belong to:

- Alloc() : Memory
- Clear() : Memory
- Copy() : Memory
- DuplicateCString() : Memory
- Exit() : Application
- Free() : Memory
- 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_lerp() : Math
- n_log2() : Math
- n_modangle() : Math
- n_rand() : Math
- n_saturate() : Math
- n_smooth() : Math
- Realloc() : Memory
- Sleep() : Timing
- Time : Timing

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:54 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

- File List
- Globals
The Nebula Device 3 File List

Here is a list of all documented files with brief descriptions:

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:/nebula3/code/foundation/foundation.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/foundation/stdneb.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/foundation/attr/valuetype.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/foundation/core/config.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/foundation/core/debug.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/foundation/core/win32/precompiled.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/foundation/core/win32/win32singleton.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/foundation/math/matrix44.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/foundation/math/scalar.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/foundation/math/d3dx9/d3dx9_scalar.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/foundation/memory/win32/win32memory.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/foundation/system/platform.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/render/timing/time.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/render/render.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/render/coregraphics/config.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/render/input/char.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/render/models/attributes.h</td>
<td></td>
</tr>
<tr>
<td>C:/nebula3/code/render/preshaders/preshaders.h</td>
<td></td>
</tr>
</tbody>
</table>

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:54 2008
Main Page
Namespaces
Data Structures
Files
Related Pages
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
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"
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages
Detailed Description

Defines the valid attribute value types as enum.

(C) 2006 Radon Labs GmbH

#include "core/types.h" #include "util/variant.h"
Namespaces

namespace Attr
Main Page
Namespaces
Data Structures
Files
Related Pages
Detailed Description

Nebula3 compiler specific defines and configuration.

(C) 2006 Radon Labs GmbH
Defines

#define NEBULA3_MEMORY_STATS (0)
#define DEFAULT_IO_SCHEME "file"
#define __attribute__(x)
Define Documentation

#define NEBULA3_MEMORY_STATS (0)

Nebula3 configuration.

#define DEFAULT_IO_SCHEME "file"

Xbox360 specifics. **Win32** specifics. Default values and pathes.

#define __attribute__(x)


The Nebula Device 3 documentation generated by **doxygen** at Tue Feb 19 12:16:39 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages
C:/nebula3/code/foundation/core/debug.h
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
## Functions

<table>
<thead>
<tr>
<th>void __cdecl void __cdecl void __cdecl void</th>
<th>n_sleep (double)</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>n_barf (const char *, const char *, int) <strong>attribute</strong>((noreturn))</td>
</tr>
<tr>
<td>void</td>
<td>n_barf2 (const char *, const char *, const char *, int) <strong>attribute</strong>((noreturn))</td>
</tr>
</tbody>
</table>
Function Documentation

```c
void __cdecl void __cdecl void __cdecl ( double sec )
void __cdecl n_sleep

Put process to sleep.
```

- 21-Dec-98 floh created

```c
void const n_barf ( char *exp,
  const char *file,
  int line )
```

This function is called by n_assert() when the assertion fails.

```c
void const n_barf2 ( char *exp,
  const char *msg,
  const char *file,
  int line )
```

This function is called by n_assert2() when the assertion fails.
Main Page
Namespaces
Data Structures
Files
Related Pages
C:/nebula3/code/foundation/core/win32/precompiled.h
File Reference
Detailed Description

Contains precompiled headers on the Win32 platform.

(C) 2007 Radon Labs GmbH

#include <windows.h> #include <process.h>
#include <shfolder.h>
#include <strsafe.h>
#include <wininet.h>
#include <winsock2.h>
#include <rpc.h>
#include <dbghelp.h>
#include <intrin.h>
#include <math.h>
#include <stdlib.h>
#include <stdarg.h>
#include <algorithm>
#include <d3d9.h>
#include <d3dx9.h>
#include <xinput.h>
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages
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"
Main Page
Namespaces
Data Structures
Files
Related Pages
Detailed Description

Frontend header for matrix classes.

(C) 2006 Radon Labs GmbH

#include "math/d3dx9/d3dx9_matrix44.h"
Main Page
Namespaces
Data Structures
Files
Related Pages
Detailed Description

Nebula's scalar datatype.

(C) 2007 Radon Labs GmbH

#include "math/d3dx9/d3dx9_scalar.h"
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages
Detailed Description

Scalar typedef and math functions for D3DX9 math functions.

(C) 2007 Radon Labs GmbH

#include "core/types.h"
Namespaces

namespace Math
<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__forceinline</td>
<td>Math::n_log2</td>
<td>(scalar f)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_iclamp</td>
<td>(int val, int minVal, int maxVal)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_fequal</td>
<td>(scalar f0, scalar f1, scalar tol)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_fless</td>
<td>(scalar f0, scalar f1, scalar tol)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_fgreater</td>
<td>(scalar f0, scalar f1, scalar tol)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_smooth</td>
<td>(scalar newVal, scalar curVal, scalar maxChange)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_clamp</td>
<td>(scalar val, scalar lower, scalar upper)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_saturate</td>
<td>(scalar val)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_rand</td>
<td>()</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_rand</td>
<td>(scalar min, scalar max)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_fchop</td>
<td>(scalar f)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_frnd</td>
<td>(scalar f)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_lerp</td>
<td>(scalar x, scalar y, scalar l)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_modangle</td>
<td>(scalar a)</td>
</tr>
<tr>
<td>__forceinline</td>
<td>Math::n_angulardistance</td>
<td>(scalar from, scalar to)</td>
</tr>
</tbody>
</table>
Detailed Description

Low level memory functions for the **Win32** platform.

(C) 2006 Radon Labs GmbH

#include "core/config.h" #include "core/debug.h"
#include "threading/interlocked.h"
Namespaces

namespace Memory
Data Structures

```c
struct Memory::MemoryStatus
```
## Functions

<table>
<thead>
<tr>
<th>__forceinline void *</th>
<th>Memory::Alloc (size_t size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>__forceinline void *</td>
<td>Memory::Realloc (void *ptr, size_t size)</td>
</tr>
<tr>
<td>__forceinline void</td>
<td>Memory::Free (void *ptr)</td>
</tr>
<tr>
<td>__forceinline void</td>
<td>Memory::Copy (const void *from, void *to, size_t numBytes)</td>
</tr>
<tr>
<td>__forceinline void</td>
<td>Memory::Clear (void *ptr, size_t numBytes)</td>
</tr>
<tr>
<td>__forceinline char *</td>
<td>Memory::DuplicateCString (const char *from)</td>
</tr>
<tr>
<td>__forceinline __cdecl</td>
<td>operator new (size_t size)</td>
</tr>
<tr>
<td>__forceinline __cdecl</td>
<td>operator new[] (size_t size)</td>
</tr>
<tr>
<td>__forceinline __cdecl</td>
<td>operator delete (void *p)</td>
</tr>
<tr>
<td>__forceinline __cdecl</td>
<td>operator delete[] (void *p)</td>
</tr>
</tbody>
</table>
Function Documentation

__forceinline
void*
__cdecl
operator
new
(size_t size)

Replacement global new operator.

__forceinline
void*
__cdecl
operator
new[]
(size_t size)

Replacement global new[] operator.

__forceinline
void
__cdecl
operator
(delete
(void * p))

Replacement global delete operator.

__forceinline
void
__cdecl
operator
(delete[]
(void * p))

Replacement global delete[] operator.
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages
C:/nebula3/code/foundation/system/platform.h
File Reference
Detailed Description

Platform-specific header

(C) 2007 Radon Labs GmbH

#include "system/config.h" #include "system/win32/platform.h"
Main Page
Namespaces
Data Structures
Files
Related Pages
Detailed Description

Typedefs for the Timing subsystem

(C) 2006 Radon Labs GmbH

#include "core/types.h"
Namespaces

namespace Timing
typedef double \texttt{Timing::Time}  
\textit{the time datatype}
Functions

```c
void Timing::Sleep (Time t)
```
<table>
<thead>
<tr>
<th>home</th>
<th>namespace list</th>
</tr>
</thead>
</table>

- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages
C:/nebula3/code/render/render.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

#include "models/modelnode.h"
Namespaces

namespace Models
Main Page
Namespaces
Data Structures
Files
Related Pages
C:/nebula3/code/render/coregraphics/con
File Reference
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
Detailed Description

A translated character code.

(C) 2006 Radon Labs GmbH

#include "core/types.h"
Namespaces

namespace Input
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages
Detailed Description

Define Models subsystem attributes.

(C) 2007 Radon Labs GmbH

#include "attr/attribute.h"
Namespaces

namespace Attr
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages
C:/nebula3/code/render/preshaders/preshad
File Reference
Detailed Description

Includes all pre-shader headers and registers class names.

(C) 2007 Radon Labs GmbH

#include "preshaders/boxfilterkernel.h" #include "preshaders/gaussianblur5x5filterkernel.h"
Namespaces

namespace PreShaders
Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- __attribute__: config.h
- DEFAULT_IO_SCHEME: config.h
- n_barf(): debug.h
- n_barf2(): debug.h
- n_sleep(): debug.h
- NEBULA3_MEMORY_STATS: config.h
- operator delete(): win32memory.h
- operator delete[](): win32memory.h
- operator new(): win32memory.h
- operator new[](): win32memory.h
The Nebula Device 3 Related Pages

Here is a list of all related documentation pages:

- Todo List

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:54 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages
Todo List

Global **Base::ShaderVariableInstanceBase::Apply** ()
  : hmm, the dynamic type switch is sort of lame...

Class **CoreGraphics::StreamAnimationLoader**
  : document file formats

Class **CoreGraphics::StreamMeshLoader**
  : document file formats

Class **Direct3D9::D3D9Shader**
  lost/reset device handling

Class **Direct3D9::D3D9ShaderInstance**
  lost/reset device handling

Class **Graphics::Cell**
  : 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)

Global **Graphics::View::ResolveVisibleLights** ()
  : currently this methods needs to go over all visible graphics entities to find the lights...

Class **IO::BinaryReader**
  convert endianess!

Class **IO::BinaryWriter**
  convert endianess!
Global `Lighting::SM30LightServer::ApplyModelEntityLights` (const `Ptr< Graphics::ModelEntity > &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 `Threading::Event` describe Event class

Class `Threading::Thread` describe Thread class

Class `Threading::ThreadLocalPtr< T >` describe ThreadLocalPtr 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 `Win32::Win32Timer` 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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>S</th>
<th>T</th>
<th>U</th>
<th>V</th>
<th>W</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractLightEntity (Lighting)</td>
<td>D3D9ShapeRenderer (Direct3D9)</td>
<td>D3D9StreamShaderLoader (Direct3D9)</td>
<td>D3D9StreamTextureLoader (Direct3D9)</td>
<td>D3D9StreamTextureSaver (Direct3D9)</td>
<td>D3D9Texture (Direct3D9)</td>
<td>D3D9TransformDevice (Base)</td>
<td>D3D9Types (Direct3D9)</td>
<td>D3D9VertexBuffer (Direct3D9)</td>
<td>D3D9VertexLayout (Direct3D9)</td>
<td>DefaultHttpRequestHandler (Http)</td>
<td>Id (Messaging)</td>
<td>ImageFileFormat (CoreGraphics)</td>
<td>IndexBuffer (CoreGraphics)</td>
<td>IndexBufferBase (Base)</td>
<td>IndexType (CoreGraphics)</td>
<td>InputEvent (Input)</td>
<td>InputFocus (GraphicsFeature)</td>
<td>InputHandler (Input)</td>
<td>InputPriority (Input)</td>
<td>InputServer (Input)</td>
<td>InputServerBase (Base)</td>
</tr>
<tr>
<td>core</td>
<td>GlobalLightEntity (Lighting)</td>
<td>ModelReader (Models)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------</td>
<td>----------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoreFeature (Game)</td>
<td>GraphicsEntity (Graphics)</td>
<td>ModelServer (Models)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CorePageHandler (Debug)</td>
<td>GraphicsEntityProxy (AsyncGraphics)</td>
<td>ModelWriter (Models)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoreServer (Core)</td>
<td>GraphicsFeatureUnitUnit (GraphicsFeature)</td>
<td>MountZipArchive (Interface)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cpu (System)</td>
<td>GraphicsServer (Graphics)</td>
<td>Mouse (Input)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPUIndexBuffer</td>
<td>GraphicsServerProxy (AsyncGraphics)</td>
<td>MouseBase (Base)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPUMemoryIndexBufferLoader</td>
<td>Guid (Util)</td>
<td>MouseButton (Input)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPUMemoryVertexBufferLoader</td>
<td>GuidAttrId (Attr)</td>
<td>MouseGripperProperty (PhysicsFeature)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPUVertexBuffer</td>
<td>Handle (AsyncGraphics)</td>
<td>MoveDirection (Messaging)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Handler (Messaging)</td>
<td>MoveFollow (BaseGameFeature)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CriticalSection (Threading)</td>
<td>HasAttachment (Messaging)</td>
<td>MoveGoto (Messaging)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MoveRotate (Messaging)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>MoveSetVelocity (Messaging)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MoveStop (Messaging)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MoveTurn (Messaging)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9DisplayDevice</td>
<td></td>
<td>HashTable (Util)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9IndexBuffer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryIndexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryVertexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RendererDevice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RenderTarget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9Shader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderInstance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderServer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9DisplayDevice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9IndexBuffer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryIndexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryVertexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RendererDevice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RenderTarget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9Shader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderInstance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderServer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9DisplayDevice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9IndexBuffer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryIndexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryVertexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RendererDevice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RenderTarget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9Shader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderInstance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderServer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9DisplayDevice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9IndexBuffer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryIndexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryVertexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RendererDevice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RenderTarget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9Shader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderInstance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderServer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9DisplayDevice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9IndexBuffer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryIndexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryVertexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RendererDevice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RenderTarget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9Shader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderInstance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderServer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9DisplayDevice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9IndexBuffer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryIndexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9MemoryVertexBufferLoader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RendererDevice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9RenderTarget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9Shader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderInstance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderServer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3D9ShaderVariable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Direct3D9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
- a -

- abs() : Math::float2
- AbstractLightEntity() : Lighting::AbstractLightEntity
- Accept() : Win32::Win32Socket
- AcceptsMessage() : Messaging::Port, Game::Entity
- AcceptsRequest() : Debug::IoPageHandler, Debug::MemoryPageHandler, Debug::ScriptingPageHandler, Debug::DisplayPageHandler, Debug::CorePageHandler, Debug::MeshPageHandler, Debug::ShaderPageHandler, Http::HttpRequestHandler, Debug::TexturePageHandler
- ActivateEntity() : BaseGameFeature::EntityManager
- ActorEntity() : Graphics::ActorEntity
- AdapterExists() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice
- AdapterInfo() : CoreGraphics::AdapterInfo
- Add() : Win32::Win32Interlocked, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Models::VisResolveContainer< TYPE >
- AddAfter() : Util::List< TYPE >
- AddArg() : Scripting::ArgsBlock
- AddAttr() : Attr::AttributeContainer, Http::HtmlPageWriter
- AddBack() : Util::List< TYPE >
- AddBatch() : Frame::FramePass
- AddBefore() : Util::List< TYPE >
- AddBlob() : Game::Entity
- AddBool() : Game::Entity
- AddCategoryAttr() : BaseGameFeature::CategoryManager
- AddChild() : Models::ModelNode,
Models::ModelNodeInstance
- AddColorBuffer() : Base::RenderTargetBase
- AddColumn() : Attr::AttributeTable
- AddDelayedJob() : BaseGameFeature::EntityManager
- AddDependency() : Graphics::View
- AddDepthStencilBuffer() : Base::RenderTargetBase
- AddDirEntry() : IO::ZipDirEntry
- AddFileEntry() : IO::ZipDirEntry
- AddFloat() : Game::Entity
- AddFloat4() : Game::Entity
- AddFramePass() : Frame::FrameShader
- AddFront() : Util::List< TYPE >
- AddGuid() : Game::Entity
- AddInt() : Game::Entity
- AddLink() : Graphics::GraphicsEntity
- AddList() : Util::List< TYPE >
- AddMatrix44() : Game::Entity
- AddPostEffect() : Frame::FrameShader
- AddPreShader() : Base::ShaderInstanceBase
- AddProperties() : BaseGameFeature::FactoryManager
- AddRef() : Core::RefCounted, Util::SimpleTree< VALUETYPE >::Node
- AddRenderTarget() : Frame::FrameShader
- AddRow() : Attr::AttributeTable
- AddStateHandler() : App::GameApplication
- AddString() : Game::Entity
- AddVariable() : Frame::FrameBatch, Frame::FramePass, Frame::FramePostEffect
- AddVisibleNodeInstance() : Models::ModelNode
- AdvanceProgress() : BaseGameFeature::LoaderServer
- all() : Math::float2
- Alloc() : Win32::Win32Heap
- ANSItoUTF8() : Util::String
- any() : Math::float2
- Append() : Util::Array< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::String
- AppendArray() : Util::Array< TYPE >
- AppendBool() : Util::String
- AppendFloat() : Util::String
- `AppendFloat4()` : `Util::String`
- `AppendInt()` : `Util::String`
- `AppendLocalPath()` : `IO::URI`
- `AppendMatrix44()` : `Util::String`
- `AppendRange()` : `Util::String`
- `Application()` : `App::Application`
- `Apply()` : `Base::ShaderVariableInstanceBase`
- `ApplyImpulseAtPos()` : `PhysicsFeature::PhysicsProperty`
- `ApplyModelEntityLights()` : `Lighting::LightServerBase`, `Lighting::SM30LightServer`
- `ApplyModelTransforms()` : `Base::TransformDeviceBase`
- `ApplyPrimitives()` : `Base::MeshBase`
- `ApplySharedState()` : `Models::ModelNode`, `Models::ShapeNode`, `Models::SkinShapeNode`, `Models::StateNode`
- `ApplyState()` : `Models::ModelNodeInstance`, `Models::StateNodeInstance`, `Models::TransformNodeInstance`
- `ApplyViewSettings()` : `Base::TransformDeviceBase`
- `AreMipMapsEnabled()` : `Base::RenderTargetBase`
- `ArgsBlock()` : `Scripting::ArgsBlock`
- `Arguments()` : `Scripting::Command`
- `ArgValue()` : `Scripting::ArgsBlock`
- `Array()` : `Util::Array<TYPE>`
- `AsBinary()` : `Win32::Win32Guid`
- `AsBool()` : `Util::String`
- `AsCharPtr()` : `Util::String`
- `AsD3D9MultiSampleType()` : `Direct3D9::D3D9Types`
- `AsD3D9PixelFormat()` : `Direct3D9::D3D9Types`
- `AsD3D9PrimitiveType()` : `Direct3D9::D3D9Types`
- `AsD3D9VertexDeclarationType()` : `Direct3D9::D3D9Types`
- `AsD3D9VertexDeclarationUsage()` : `Direct3D9::D3D9Types`
- `AsD3DXImageFileFormat()` : `Direct3D9::D3D9Types`
- `AsFloat()` : `Util::String`
- `AsFloat4()` : `Util::String`
- `AsInt()` : `Util::String`
- `AsMatrix44()` : `Util::String`
- `AsNebulaPixelFormat()` : `Direct3D9::D3D9Types`
- `Assign()` : `IO::Assign`
- `AsString()` : `Win32::Win32Guid`, `Util::FourCC`, `IO::MediaType`, `IO::URI`
- `AsUInt()` : `Util::FourCC`
- `AsyncGraphicsHandler()` : `AsyncGraphics::AsyncGraphicsHandler`
- `AsyncGraphicsInterface()` : `AsyncGraphics::AsyncGraphicsInterface`
- `AsyncHttpInterface()` : `AsyncHttp::AsyncHttpInterface`
- `AsyncPort()` : `Messaging::AsyncPort`
- `AsyncRenderApplication()` : `App::AsyncRenderApplication`
- `AsyncViewerApplication()` : `App::AsyncViewerApplication`
- `At()` : `Util::Array< TYPE >`, `Util::FixedTable< TYPE >`
- `Atom()` : `Util::Atom< TYPE >`
- `AttachChildCell()` : `Graphics::Cell`
- `AttachDisplayEventHandler()` : `AsyncGraphics::DisplayProxy`
- `AttachEntity()` : `Graphics::Stage`, `BaseGameFeature::EntityManager`, `Graphics::Cell`
- `AttachEntityLoader()` : `BaseGameFeature::LoaderServer`
- `AttachEventHandler()` : `Base::RenderDeviceBase`, `Base::DisplayDeviceBase`
- `AttachGameFeature()` : `Game::GameServer`
- `AttachHandler()` : `Messaging::Port`, `Messaging::AsyncPort`, `IO::Console`
- `AttachInputHandler()` : `Base::InputServerBase`
- `AttachManager()` : `Game::FeatureUnit`
- `AttachMapper()` : `Resources::ResourceManager`
- `AttachNode()` : `Models::Model`
- `AttachPort()` : `Messaging::Dispatcher`
- `AttachRenderEventHandler()` : `AsyncGraphics::DisplayProxy`
- `AttachRequestHandler()` : `Http::HttpServer`
- `AttachVisibleLight()` : `Lighting::LightServerBase`
- `AttachVisibleModelInstance()` : `Models::VisResolver`
- `Attribute()` : `Attr::Attribute`
- `AttributeContainer()` : `Attr::AttributeContainer`
- `AttributeDefinition()` : `Attr::AttributeDefinition< VALUETYPE, TYPE >`
- `AttributeDefinitionBase()` : `Attr::AttributeDefinitionBase`
- `AttributeTable()` : `Attr::AttributeTable`
- `AttrId()` : `Attr::AttrId`
- coreFeature : `App::GameApplication`
- gameServer : `App::GameApplication`
- MaxNumColorBuffers : `Base::RenderTargetBase`
- NumSplits : `Lighting::PSSMUtil`
- requestedState : `App::GameApplication`
- sampledCurves : `Models::ParticleSystemNode`
- Creator: Core::Rtti
- EntityId: Game::Entity
- Iterator: Util::Array< TYPE >, Util::FixedArray< TYPE >
- Mask: CoreGraphics::ShaderFeature
- Name: Base::ShaderVariableBase, CoreGraphics::ShaderFeature
- Position: IO::Stream
- Semantic: Base::ShaderVariableBase
- Year: Base::CalendarTimeBase
- a -
- AccessMode : `IO::Stream`
- AccessPattern : `IO::Stream`
- AccessType : `CoreGraphics::VertexComponent`

- Button : `Base::GamePadBase`

- CallbackType : `Game::Property`
- Code : `Http::HtmlElement`, `Http::HttpStatus`, `Models::ModelNodeType`, `CoreGraphics::DisplayEvent`, `Input::Key`, `Frame::SortingMode`, `Frame::LightingMode`, `CoreGraphics::RenderEvent`, `CoreGraphics::PrimitiveTopology`, `CoreGraphics::PixelFormat`, `CoreGraphics::IndexType`, `Input::MouseButton`, `CoreGraphics::ImageFileFormat`, `CoreGraphics::BatchType`, `CoreGraphics::AntiAliasQuality`, `CoreGraphics::Adapter`, `Http::HttpMethod`
- CoreId : `System::Cpu`
- CubeFace : `Base::TextureBase`

- DelayedJobType : `BaseGameFeature::EntityManager`

- ErrorCode : `Win32::Win32Socket`

- Format : `CoreGraphics::VertexComponent`

- LinkType : `Graphics::GraphicsEntity`
- m -
  - Month : `Base::CalendarTimeBase`

- p -
  - Priority : `Win32::Win32Thread`, `Resources::ManagedResource`
  - Protocol : `Win32::Win32Socket`

- r -
  - RootKey : `System::Win32Registry`

- s -
  - SeekOrigin : `IO::Stream`
  - SemanticName : `CoreGraphics::VertexComponent`
  - SetupMode : `App::GameStateHandler`
  - ShaderParamBindMode : `Base::ShaderServerBase`
  - ShapeType : `Base::ShapeRendererBase`
  - State : `Resources::Resource`

- t -
  - Type : `Util::Variant`, `Input::InputEvent`, `Graphics::GraphicsEntity`, `Base::TextureBase`, `Base::ShaderVariableBase`

- u -
  - Usage : `Base::ResourceBase`

- w -
  - Weekday : `Base::CalendarTimeBase`
operator!= : Win32::Win32FileTime, Util::String
operator< : Util::String, Win32::Win32FileTime
operator<= : Util::String
operator== : Util::String, Win32::Win32FileTime
operator> : Win32::Win32FileTime, Util::String
operator>= : Util::String
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- Back() : Util::Array< TYPE > , Util::List< TYPE > , Util::SimpleTree< VALUETYPE >::Node
- Base64Writer() : Http::Base64Writer
- bbox() : Math::bbox
- Begin() : Http::HtmlPageWriter , Util::Array< TYPE > , Util::Crc , Util::FixedArray< TYPE > , Util::List< TYPE > , Base::ShaderInstanceBase , Base::ShapeRendererBase , Direct3D9::D3D9ShaderInstance
- begin_extend() : Math::bbox
- BeginAddCategoryAttrs() :
  BaseGameFeature::CategoryManager
- BeginAddColumns() : Attr::AttributeTable
- BeginAttachVisibleLights() : Lighting::LightServerBase
- BeginBatch() : Base::RenderDeviceBase , Base::RenderTargetBase
- BeginCapture() : Base::KeyboardBase , Base::MouseBase , Input::InputHandler
- BeginClips() : Models::CharacterNode
- BeginFragments() : Models::SkinShapeNode
- BeginFrame() : Base::RenderDeviceBase , Direct3D9::D3D9RenderDevice , Base::InputServerBase , Lighting::LightServerBase
- BeginJointPalette() : Models::SkinShapeNode
- BeginJoints() : Models::CharacterNode
- BeginModel() : Models::BinaryModelWriter , Models::ModelWriter , Models::XmlModelWriter
BeginModelNode() : Models::BinaryModelWriter,
Models::ModelWriter, Models::XmlModelWriter
BeginNode() : IO::XmlWriter
BeginPass() : Direct3D9::D3D9RenderTarget,
Base::RenderDeviceBase, Base::RenderTargetBase,
Base::ShaderInstanceBase, Direct3D9::D3D9ShaderInstance
BeginResolve() : Models::VisResolver
BeginVariations() : Models::CharacterNode
BinaryModelReader() : Models::BinaryModelReader
BinaryModelWriter() : Models::BinaryModelWriter
BinaryReader() : IO::BinaryReader
BinarySearchIndex() : Util::Array< TYPE >, Util::FixedArray<
TYPE >
BinaryWriter() : IO::BinaryWriter
Bind() : Win32::Win32Socket
Blob() : Util::Blob
BlobAttrId() : Attr::BlobAttrId
BoolAttrId() : Attr::BoolAttrId
BuildSignature() : Base::VertexLayoutBase
BuildStage() : Graphics::SimpleStageBuilder,
Graphics::StageBuilder, Graphics::QuadtreeStageBuilder
Button : Base::GamePadBase
ButtonDoubleClicked() : Base::MouseBase
ButtonDown() : Base::MouseBase, Base::GamePadBase
ButtonPressed() : Base::GamePadBase, Base::MouseBase
ButtonUp() : Base::GamePadBase, Base::MouseBase
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- C -

- CalendarTimeBase() : Base::CalendarTimeBase
- CallbackType : Game::Property
- CameraDistance() : GraphicsFeature::CameraDistance
- CameraEntity() : Graphics::CameraEntity
- CameraEntityProxy() : AsyncGraphics::CameraEntityProxy
- CameraFocus() : GraphicsFeature::CameraFocus
- CameraOrbit() : GraphicsFeature::CameraOrbit
- CanBeMapped() : IO::FileStream, IO::MemoryStream, IO::Stream, IO::ZipFileStream
- Cancel() : Messaging::AsyncPort
- CanCreate() : Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
- CanLoadAsync() : Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, Models::StreamModelLoader, Resources::ResourceLoader, Direct3D9::D3D9StreamShaderLoader
- CanRead() : IO::FileStream, IO::MemoryStream, IO::Stream, IO::ZipFileStream
- CanSeek() : IO::FileStream, IO::MemoryStream, IO::Stream, IO::ZipFileStream
- CanWrite() : IO::ZipFileStream, IO::FileStream, IO::MemoryStream, IO::Stream
- Capacity() : Util::Array< TYPE >, Util::HashTable< KEYTYPE, VALUETYPE >
- Category() : BaseGameFeature::CategoryManager::Category
BaseGameFeature::CategoryManager::Entry
- CategoryManager() : BaseGameFeature::CategoryManager
- Cell() : Graphics::Cell
- center() : Math::bbox
- CharacterNode() : Models::CharacterNode
- CharacterNodeInstance() : Models::CharacterNodeInstance
- CheckFileExtension() : Util::String
- CheckId() : Messaging::Message
- CheckValidCharSet() : Util::String
- Child() : Util::SimpleTree< VALUETYPE >::Node
- ClassExists() : Core::Factory
- Cleanup() : Base::ShaderInstanceBase, Direct3D9::D3D9ShaderInstance, BaseGameFeature::EntityManager
- CleanupGameFeatures() : App::GameApplication
- CleanupStateHandlers() : App::GameApplication
- Clear() : Util::Dictionary< KEYTYPE, VALUETYPE >, Util::FixedArray< TYPE >, Util::FixedTable< TYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::List< TYPE >, Util::Proxy< TYPE >, Util::Queue< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::Stack< TYPE >, Util::String, Util::Variant, Resources::ManagedResource, Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::MediaType, IO::URI, Scripting::ArgsBlock, Threading::SafePriorityQueue< PRITYPE, TYPE >, Threading::SafeQueue< TYPE >, Util::Array< TYPE >, Util::Atom< TYPE >
- ClearAssign() : IO::IoServer
- ClearCapture() : Base::InputServerBase
- ClearDataPtr() : Util::QuadTree< TYPE >::Node
- ClearDeletedRowsFlags() : Attr::AttributeTable
- ClearDynamicAttributes() : Attr::AttributeDefinitionBase
- ClearEntity() : Game::Property
- ClearEnvEntity() : BaseGameFeature::EnvEntityManager
- ClearError() : Scripting::ScriptServer
- ClearFeatureBits() : Base::ShaderServerBase
- ClearKeyboardCapture() : Base::InputServerBase
- ClearLinks() : Graphics::GraphicsEntity
- ClearMouseCapture() : Base::InputServerBase
- ClearNewRowFlags() : Attr::AttributeTable
- ClearRenderStats() : Resources::ManagedResource
- clipstatus() : Math::bbox, Math::sphere
- Close() : Base::TransformDeviceBase, Direct3D9::D3D9RenderDevice, Direct3D9::D3D9ShaderServer, Direct3D9::D3D9ShapeRenderer, CoreGraphics::VertexLayoutServer, Win32::Win32DisplayDevice, Frame::FrameServer, Graphics::GraphicsServer, Base::InputServerBase, Win32::Win32InputServer, Lighting::LightServerBase, Lighting::SM30LightServer, Lighting::SM30ShadowServer, Models::BinaryModelReader, Models::ModelReader, Models::N2ModelReader, Models::XmlModelReader, Models::ModelServer, Models::BinaryModelWriter, Models::ModelWriter, Models::XmlModelWriter, Models::VisResolver, Resources::ResourceManager, Resources::SharedResourceServer, App::GameApplication, BaseGameFeature::LoaderServer, Game::GameServer, App::Application, App::ConsoleApplication, Core::CoreServer, Http::HtmlPageWriter, Http::HttpServer, IO::BinaryReader, IO::BinaryWriter, IO::Console, IO::ConsoleHandler, IO::FileStream, IO::MemoryStream, IO::Stream, IO::StreamReader, IO::StreamWriter, IO::XmlReader, IO::XmlWriter, IO::ZipFileStream, IO::ZipArchive, Messaging::AsyncPort, Messaging::Handler, Net::TcpServer, Win32::Win32Socket, Scripting::LuaServer, Scripting::ScriptServer, App::AsyncRenderApplication, App::AsyncViewerApplication, App::RenderApplication, App::ViewerApplication, AsyncGraphics::AsyncGraphicsHandler, AsyncGraphics::DisplayProxy, AsyncGraphics::GraphicsServerProxy, Base::DisplayDeviceBase, Base::RenderDeviceBase, Base::ShaderServerBase, Base::ShapeRendererBase
- CloseDInputMouse() : Win32::Win32InputServer
- CloseFile() : Win32::Win32FSWrapper
- CloseProgressIndicator() : BaseGameFeature::LoaderServer
- closestpoint() : Math::line
- CloseWindow() : `Win32::Win32DisplayDevice`
- CommandLineArgs() : `Util::CmdLineArgs`
- Code : `Models::ModelNodeType`, `Frame::LightingMode`, `CoreGraphics::IndexType`, `Http::HtmlElement`, `Http::HttpMethod`, `Http::HttpStatus`, `CoreGraphics::Adapter`, `CoreGraphics::BatchType`, `CoreGraphics::DisplayEvent`, `CoreGraphics::ImageFileFormat`, `CoreGraphics::PixelFormat`, `CoreGraphics::PrimitiveTopology`, `CoreGraphics::RenderEvent`, `CoreGraphics::AntiAliasQuality`, `Frame::SortingMode`, `Input::Key`, `Input::MouseButton`
- Column() : `Util::QuadTree< TYPE >::Node`
- Command() : `Scripting::Command`
- Commit() : `Direct3D9::D3D9ShaderInstance`, `Base::ShaderInstanceBase`
- CommitChangesToDatabase() : `BaseGameFeature::CategoryManager`
- Compute() : `Util::Crc`, `Lighting::PSSMUtil`
- ComputeAbsMousePos() : `Win32::Win32DisplayDevice`
- ComputeAdjustedWindowRect() : `Win32::Win32DisplayDevice`
- ComputeClipStatus() : `Lighting::GlobalLightEntity`, `Graphics::CameraEntity`, `Graphics::GraphicsEntity`, `Lighting::SpotLightEntity`
- ComputeFileCrc() : `IO::IoServer`
- ComputeMouseWorldRay() : `BaseGameFeature::EnvQueryManager`
- ComputeNormMousePos() : `Win32::Win32DisplayDevice`
- ComputeWorldMouseRay() : `Graphics::GraphicsServer`
- Concatenate() : `Util::String`
- Connect() : `Net::TcpClient`, `Net::TcpClientConnection`, `Win32::Win32Socket`
- Console() : `IO::Console`
- ConsoleApplication() : `App::ConsoleApplication`
- ConsoleHandler() : `IO::ConsoleHandler`
- contains() : `Math::bbox`
- Contains() : `Util::Dictionary< KEYTYPE, VALUETYPE >`, `Util::HashTable< KEYTYPE, VALUETYPE >`
- contains() : `Math::bbox`
- Contains() : `Util::Queue< TYPE >`, `Util::Stack< TYPE >`
- ContainsCharFromSet() : Util::String
- Content() : Util::HashTable< KEYTYPE, VALUETYPE >
- ConvertBackslashes() : Util::String
- ConvertDirToZipURIIfExists() : IO::ZipFileSystem
- ConvertDouble() : System::ByteOrder
- ConvertFileToZipURIIfExists() : IO::ZipFileSystem
- ConvertFloat() : System::ByteOrder
- ConvertFloat4() : System::ByteOrder
- ConvertInt() : System::ByteOrder
- ConvertMatrix44() : System::ByteOrder
- ConvertShort() : System::ByteOrder
- ConvertToPathInZipArchive() : IO::ZipArchive
- ConvertToZipURI() : IO::ZipArchive
- ConvertUInt() : System::ByteOrder
- ConvertUShort() : System::ByteOrder
- CopyExtRow() : Attr::AttributeTable
- CopyFile() : IO::IoServer, Interface::CopyFile
- CopyRow() : Attr::AttributeTable
- coreFeature : App::GameApplication
- CoreId : System::Cpu
- CorePageHandler() : Debug::CorePageHandler
- CoreServer() : Core::CoreServer
- corner_point() : Math::bbox
- CPUIndexBuffer() : CoreGraphics::CPUIndexBuffer
- CPUVertexBuffer() : CoreGraphics::CPUVertexBuffer
- Crc() : Util::Crc
- Create() : Core::Factory, Core::Rtti
- CreateClone() : Graphics::GraphicsEntity
- CreateDirectory() : IO::IoServer, Win32::Win32FSWrapper
- 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`
- `CreateEnvEntity()`: `BaseGameFeature::EnvEntityManager`
- `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::ShapeNode`, `Models::ModelNode`, `Models::StateNode`, `Models::TransformNode`, `Models::ParticleSystemNode`, `Models::CharacterNode`, `Models::SkinShapeNode`
- `CreateProperty()`: `BaseGameFeature::FactoryManager`
- `CreateShaderInstance()`: `Base::ShaderBase`, `Base::ShaderServerBase`
- `CreateShaderVariableInstance()`: `Models::StateNodeInstance`
- `CreateSharedResource()`: `Resources::SharedResourceServer`
- `CreateSharedVertexLayout()`: `CoreGraphics::VertexLayoutServer`
- `CreateStage()`: `Graphics::GraphicsServer`
- `CreateStageProxy()`: `AsyncGraphics::GraphicsServerProxy`
- `CreateStream()`: `IO::IoServer`
- `CreateUserProfile()`: `BaseGameFeature::LoaderServer`
- `CreateView()`: `Graphics::GraphicsServer`
- `CreateViewProxy()`: `AsyncGraphics::GraphicsServerProxy`
- `Creator`: `Core::Rtti`
- `CubeFace`: `Base::TextureBase`
The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- d -

- D3D9DisplayDevice() : Direct3D9::D3D9DisplayDevice
- D3D9IndexBuffer() : Direct3D9::D3D9IndexBuffer
- D3D9RenderDevice() : Direct3D9::D3D9RenderDevice
- D3D9RenderTarget() : Direct3D9::D3D9RenderTarget
- D3D9Shader() : Direct3D9::D3D9Shader
- D3D9ShaderInstance() : Direct3D9::D3D9ShaderInstance
- D3D9ShaderServer() : Direct3D9::D3D9ShaderServer
- D3D9ShaderVariable() : Direct3D9::D3D9ShaderVariable
- D3D9ShaderVariation() : Direct3D9::D3D9ShaderVariation
- D3D9ShapeRenderer() : Direct3D9::D3D9ShapeRenderer
- D3D9Texture() : Direct3D9::D3D9Texture
- D3D9VertexBuffer() : Direct3D9::D3D9VertexBuffer
- D3D9VertexLayout() : Direct3D9::D3D9VertexLayout
- DeactivateEntity() : BaseGameFeature::EntityManager
- DebugOut() : IO::Console, IO::ConsoleHandler, Win32::Win32ConsoleHandler, Win32::SysFunc
- Decode() : Interface::CopyFile, Interface::IOMessage, Interface::ReadStream, Interface::WriteStream, Messaging::Message
- DecrClientCount() : Resources::ManagedResource
- Decrement() : Win32::Win32Interlocked
- DecrUseCount() : Resources::Resource
- DelayedJob() : BaseGameFeature::EntityManager::DelayedJob
- DelayedJobType : BaseGameFeature::EntityManager
- Delete() : System::Win32Registry
- DeleteAllRows() : Attr::AttributeTable
- DeleteDirectory() : IO::IoServer, Win32::Win32FSWrapper
- DeleteEntity() : BaseGameFeature::EntityManager
- DeleteEntityImmediate() : BaseGameFeature::EntityManager
- DeleteEnvEntity() : BaseGameFeature::EnvEntityManager
- DeleteFile() : IO::IoServer, Win32::Win32FSWrapper
- DeleteInstance() : BaseGameFeature::CategoryManager
- DeleteLevel() : BaseGameFeature::CategoryManager
- DeleteProfile() : BaseGameFeature::UserProfile
- DeleteRow() : Attr::AttributeTable
- Dequeue() : Threading::SafePriorityQueue<PRITYPE, TYPE>, Threading::SafeQueue<TYPE>, Util::Queue<TYPE>
- DequeueAll() : Threading::SafeQueue<TYPE>
- Destroy() : Attr::AttributeDefinitionBase, Core::Factory
diagonal_size() : Math::bbox
- Dictionary() : Util::Dictionary<KEYTYPE, VALUETYPE>
- Difference() : Util::Array<TYPE>
- DirectoryExists() : IO::IoServer, Win32::Win32FSWrapper
- DisablePhysics() : PhysicsFeature::PhysicsProperty
- Discard() : Frame::FramePostEffect, Frame::FrameShader, Models::ModellInstance, Base::ShaderInstanceBase, Frame::FramePass, AsyncGraphics::GraphicsEntityProxy, Base::RenderTargetBase, Base::VertexLayoutBase, Direct3D9::D3D9RenderTarget, Frame::FrameBatch
- DiscardAllStageProxies():
  AsyncGraphics::GraphicsServerProxy
- DiscardAllStages() : Graphics::GraphicsServer
- DiscardAllViewProxies() :
  AsyncGraphics::GraphicsServerProxy
- DiscardAllViews() : Graphics::GraphicsServer
- DiscardManagedModel() : Models::ModelServer
- DiscardManagedResource() : Resources::ResourceManager
- DiscardShaderInstance() : Base::ShaderBase
- DiscardStage() : Graphics::GraphicsServer
- DiscardStageProxy() : AsyncGraphics::GraphicsServerProxy
- DiscardView() : Graphics::GraphicsServer
- DiscardViewProxy() : AsyncGraphics::GraphicsServerProxy
- Disconnect() : Net::TcpClient
- Dispatcher() : Messaging::Dispatcher
- DisplayDevice() : CoreGraphics::DisplayDevice
- DisplayDeviceBase() : Base::DisplayDeviceBase
- DisplayEvent() : CoreGraphics::DisplayEvent
- DisplayEventHandler() : CoreGraphics::DisplayEventHandler
- DisplayMode() : CoreGraphics::DisplayMode
- DisplayPageHandler() : Debug::DisplayPageHandler
- DisplayProxy() : AsyncGraphics::DisplayProxy
- distance() : Math::line
- DoStateTransition() : App::GameApplication
- DoWork() : Win32::Win32Thread, Messaging::Handler, AsyncGraphics::AsyncGraphicsHandler
- Draw() : Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
- DrawIndexedPrimitives() : Base::ShapeRendererBase, Direct3D9::D3D9ShapeRenderer
- DrawPrimitives() : Base::ShapeRendererBase, Direct3D9::D3D9ShapeRenderer
- DrawShape() : Direct3D9::D3D9ShapeRenderer, Base::ShapeRendererBase
- DumpLeaks() : Core::RefCountedList
- DumpRefCountingLeaks() : Core::RefCounted
- DuplicateLevel() : BaseGameFeature::CategoryManager
- DynamicMeshResourceLoader() : Resources::DynamicMeshResourceLoader
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](http://www.example.com)
- EmitWakeupSignal() : [Win32::Win32Thread](http://www.example.com)
- EnablePhysics() : [PhysicsFeature::PhysicsProperty](http://www.example.com)
- Encode() : [Interface::CopyFile](http://www.example.com), [Messaging::Message](http://www.example.com), [Interface::IOMessage](http://www.example.com), [Interface::ReadStream](http://www.example.com), [Interface::WriteStream](http://www.example.com)
- End() : [Base::ShapeRendererBase](http://www.example.com), [Direct3D9::D3D9ShaderInstance](http://www.example.com)
- end() : [Math::line](http://www.example.com)
- End() : [Http::HtmlPageWriter](http://www.example.com), [Util::Array< TYPE >](http://www.example.com), [Util::Crc](http://www.example.com), [Util::FixedArray< TYPE >](http://www.example.com), [Util::List< TYPE >](http://www.example.com), [Base::ShaderInstanceBase](http://www.example.com)
- end_extend() : [Math::bbox](http://www.example.com)
- EndAddCategoryAttrs() : [BaseGameFeature::CategoryManager](http://www.example.com)
- EndAddColumns() : [Attr::AttributeTable](http://www.example.com)
- EndAttachVisibleLights() : [Lighting::LightServerBase](http://www.example.com)
- EndBatch() : [Base::RenderDeviceBase](http://www.example.com), [Base::RenderTargetBase](http://www.example.com), [Direct3D9::D3D9RenderDevice](http://www.example.com), [Base::InputServerBase](http://www.example.com)
- EndCapture() : [Base::KeyboardBase](http://www.example.com), [Base::MouseBase](http://www.example.com), [Input::InputHandler](http://www.example.com)
- EndClips() : [Models::CharacterNode](http://www.example.com)
- EndFragments() : [Models::SkinShapeNode](http://www.example.com)
- EndFrame() : [Lighting::LightServerBase](http://www.example.com), [Base::RenderDeviceBase](http://www.example.com), [Direct3D9::D3D9RenderDevice](http://www.example.com), [Base::InputServerBase](http://www.example.com)
- EndJointPalette() : [Models::SkinShapeNode](http://www.example.com)
- EndJoints() : Models::CharacterNode
- EndModel() : Models::BinaryModelWriter, Models::ModelWriter, Models::XmlModelWriter
- EndModelNode() : Models::BinaryModelWriter, Models::ModelWriter, Models::XmlModelWriter
- EndNode() : IO::XmlWriter
- EndPass() : Base::RenderDeviceBase, Base::RenderTargetBase, Base::ShaderInstanceBase, Direct3D9::D3D9RenderDevice, Direct3D9::D3D9RenderTarget, Direct3D9::D3D9ShaderInstance
- EndResolve() : Models::VisResolver
- EndVariations() : Models::CharacterNode
- Enqueue() : Threading::SafeQueue< TYPE >, Util::Queue< TYPE >
- Enter() : Win32::Win32CriticalSection
- Entity() : Game::Entity
- EntityId : Game::Entity
- EntityIsInActiveLayer() : BaseGameFeature::EntityLoaderBase
- EntityLoaderBase() : BaseGameFeature::EntityLoaderBase
- EntityManager() : BaseGameFeature::EntityManager
- Entry() : BaseGameFeature::CategoryManager::Entry
- EnumProfiles() : BaseGameFeature::UserProfile
- EnvEntityExists() : BaseGameFeature::EnvEntityManager
- EnvEntityManager() : BaseGameFeature::EnvEntityManager
- EnvQueryManager() : BaseGameFeature::EnvQueryManager
- Eof() : IO::StreamReader, IO::FileStream, IO::MemoryStream, IO::Stream, Win32::Win32FSWrapper, IO::ZipFileStream
- Erase() : Util::Array< TYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::SimpleTree< VALUETYPE >::Node
- EraseAtIndex() : Util::Dictionary< KEYTYPE, VALUETYPE >
- EraseIndex() : Util::Array< TYPE >
- EraseMatchingElements() : Threading::SafePriorityQueue< PRITYPE, TYPE >, Threading::SafeQueue< TYPE >
- Error() : Win32::Win32ConsoleHandler, IO::Console, IO::ConsoleHandler, IO::Console, Win32::SysFunc
- ErrorCode : Win32::Win32Socket
- Eval() : Scripting::LuaServer, Scripting::ScriptServer
- EvalScript() : `Scripting::ScriptServer`
- EvaluateSkeleton() : `Graphics::ActorEntity`
- EvaluateVariation() : `Models::CharacterNode`
- Exists() : `System::Win32Registry`
- ExistsEntityByAttr() : `BaseGameFeature::EntityManager`
- ExistsEntityByUniqueId() : `BaseGameFeature::EntityManager`
- Exit() : `Win32::SysFunc`, `App::Application`
- extend() : `Math::bbox`
- extents() : `Math::bbox`
- ExtractDirName() : `Util::String`
- ExtractFileName() : `Util::String`
- ExtractFromUri() : `Win32::Win32IpAddress`
- ExtractLastDirName() : `Util::String`
- ExtractRange() : `Util::String`
- ExtractToEnd() : `Util::String`
- ExtractToLastSlash() : `Util::String`
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
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**
- FadeAnimation() : **Graphics::ActorEntity**
- FadeAnimationMix() : **Graphics::ActorEntity**
- FadeRunningAnimationsOut() : **Graphics::ActorEntity**
- FeatureMaskToString() : **Base::ShaderServerBase**
- FeatureStringToMask() : **Base::ShaderServerBase**
- FeatureUnit() : **Game::FeatureUnit**
- FileExists() : **IO::IoServer, Win32::Win32FSWrapper**
- FileStream() : **IO::FileStream**
- FileTimeToLocalTime() : **Base::CalendarTimeBase, Win32::Win32CalendarTime**
- FileTimeToSystemTime() : **Win32::Win32CalendarTime, Base::CalendarTimeBase**
- Fill() : **Util::Array< TYPE >, Util::FixedArray< TYPE >**
- FillModel() : **Models::BinaryModelReader, Models::ModelReader, Models::N2ModelReader, Models::XmlModelReader**
- Find() : **Util::Array< TYPE >, Util::FixedArray< TYPE >, Util::List< TYPE >, Util::SimpleTree< VALUETYPE >::Node**
- FindBluePrint() : **BaseGameFeature::FactoryManager**
- FindByFourCC() : **Attr::AttributeDefinitionBase**
- FindByName() : **Attr::AttributeDefinitionBase**
- FindCharIndex() : **Util::String**
- FindComponent() : **Base::VertexLayoutBase**
- FindContainmentNode() : **Util::QuadTree< TYPE >::Node**
- FindDirEntry() : **IO::ZipArchive, IO::ZipDirEntry**
- **FindFileEntry()**: `IO::ZipArchive`, `IO::ZipDirEntry`
- **FindIndex()**: `Util::Array<TYPE>`, `Util::Dictionary<KEYTYPE, VALUETYPE>`, `Util::FixedArray<TYPE>`
- **FindInstances()**: `BaseGameFeature::CategoryManager`
- **FindProperty()**: `Game::Entity`
- **FindRowIndexByAttr()**: `Attr::AttributeTable`
- **FindRowIndexByAttrs()**: `Attr::AttributeTable`
- **FindRowIndicesByAttr()**: `Attr::AttributeTable`
- **FindRowIndicesByAttrs()**: `Attr::AttributeTable`
- **FindStateHandlerByName()**: `App::GameApplication`
- **FindStringIndex()**: `Util::String`
- **FindTemplate()**: `BaseGameFeature::CategoryManager`
- **FindTemplateByAttr()**: `BaseGameFeature::CategoryManager`
- **FindZipArchive()**: `IO::ZipFileSystem`
- **FindZipArchiveWithDir()**: `IO::ZipFileSystem`
- **FindZipArchiveWithFile()**: `IO::ZipFileSystem`
- **FixedArray()**: `Util::FixedArray<TYPE>`
- **FixedTable()**: `Util::FixedTable<TYPE>`
- **float2()**: `Math::float2`
- **Float4AttrId()**: `Attr::Float4AttrId`
- **Flush()**: `IO::FileStream`, `IO::Stream`, `Win32::Win32FSWrapper`, `Messaging::AsyncPort`
- **FocusManager()**: `BaseGameFeature::FocusManager`
- **Format**: `CoreGraphics::VertexComponent`, `Base::CalendarTimeBase`, `Util::String`
- **FormatArgList()**: `Util::String`
- **FormatToString()**: `CoreGraphics::VertexComponent`
- **FourCC()**: `Util::FourCC`
- **Fragment()**: `IO::URI`
- **FrameBatch()**: `Frame::FrameBatch`
- **FramePass()**: `Frame::FramePass`
- **FramePostEffect()**: `Frame::FramePostEffect`
- **FrameServer()**: `Frame::FrameServer`
- **FrameShader()**: `Frame::FrameShader`
- **Free()**: `Win32::Win32Heap`
- **FromBinary()**: `Win32::Win32Guid`
- **FromBool()**: `Util::String`
- **FromFloat()**: `Util::String`
- FromFloat4() : `Util::String`
- FromInt() : `Util::String`
- FromMatrix44() : `Util::String`
- FromMediaType() : `CoreGraphics::ImageFileFormat`
- FromString() : `CoreGraphics::AntiAliasQuality`, `Http::HttpStatus`, `CoreGraphics::PrimitiveTopology`, `Input::Key`, `Win32::Win32Guid`, `CoreGraphics::BatchType`, `Frame::SortingMode`, `Input::MouseButton`, `Models::ModelNodeType`, `CoreGraphics::PixelFormat`, `Frame::LightingMode`, `CoreGraphics::ImageFileFormat`, `Http::HttpMethod`, `CoreGraphics::Adapter`, `CoreGraphics::IndexType`, `Util::FourCC`
- Front() : `Util::SimpleTree< VALUETYPE >::Node`, `Util::List< TYPE >`, `Util::Array< TYPE >`
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
- GamePadBase() : Base::GamePadBase
- gameServer : App::GameApplication
- GameServer() : Game::GameServer
- GameStateHandler() : App::GameStateHandler
- ge() : Math::float2
- gen() : Math::noise
- Generate() : Win32::Win32Guid
- GenerateMipLevels() : Base::RenderTargetBase, Direct3D9::D3D9RenderTarget
- Get() : Models::VisResolveContainer< TYPE >
- get_cartesian() : Math::polar
- get_clipplanes() : Math::bbox
- GetAbsMousePos() : CoreGraphics::DisplayEvent, Input::InputEvent
- GetAcceptedMessages() : Messaging::Port
- GetAccess() : Base::ResourceBase
- GetAccessMode() : Attr::Attribute, Attr::AttributeDefinitionBase, Attr::AttrId, IO::Stream
- GetAccessPattern() : IO::Stream
- GetAccessType() : CoreGraphics::VertexComponent
- GetActiveShaderInstance() : Base::ShaderServerBase
- GetActiveVariation() : Base::ShaderInstanceBase
- GetActivityDistance() : Models::ParticleSystemNode
- GetAdapter() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
GetAdapterInfo() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice
GetAddress() : Net::TcpServer, Win32::Win32Socket
 GetAllAssigns() : IO::IoServer
 GetAllAttrIds() : Attr::AttrId
 GetAllRegisteredUriSchemes() : IO::IoServer
 GetAllShaderInstances() : Base::ShaderBase
 GetAllShaders() : Base::ShaderServerBase
 GetAnim() : Models::CharacterNode
 GetAnimationMapping() : Graphics::ActorEntity
 GetAnimClipScheduler() : Graphics::ActorEntity
 GetAntiAliasQuality() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase, Base::RenderTargetBase
 GetAppName() : App::Application, Core::CoreServer
 GetArgName() : Scripting::ArgsBlock
 GetArguments() : Scripting::Command
 GetArgValue() : Scripting::ArgsBlock
 GetAspect() : AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
 GetAspectRatio() : CoreGraphics::DisplayMode
 GetAssign() : IO::IoServer
 GetAtomTableSize() : Util::Atom<TYPE>
 GetAttr() : Attr::AttributeContainer, Models::Model, Models::ModelNode, Game::Entity
 GetAttributes() : Graphics::StageBuilder
 GetAttrId() : Attr::Attribute
 GetAttrs() : Models::Model, Models::ModelNode, Attr::AttributeContainer, IO::XmlReader
 GetAttrTable() : Game::Entity
 GetAttrTableRowIndex() : Game::Entity
 GetAvailableDisplayModes() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice
 GetAxisValue() : Base::GamePadBase
 GetBackLightColor() : Lighting::GlobalLightEntity
 GetBaseAnimation() : Graphics::ActorEntity
 GetBaseAnimationDuration() : Graphics::ActorEntity
 GetBaseClip() : Graphics::ActorEntity
 GetBaseIndex() : CoreGraphics::PrimitiveGroup
 GetBaseVertex() : CoreGraphics::PrimitiveGroup
- GetBatchByIndex() : Frame::FramePass
- GetBillboardOrientation() : Models::ParticleSystemNode
- GetBinDirectory() : Win32::Win32FSWrapper
- GetBlob() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::GlobalAttrsManager, Game::Entity
- GetBlobArray() : Util::Variant
- GetBlobDefValue() : Attr::AttrId
- GetBlocking() : Win32::Win32Socket
- GetBool() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlReader, Util::CommandLineArgs, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- GetBlobArray() : Util::Variant
- GetBlobAtIndex() : Util::CommandLineArgs
- GetBlobDefValue() : Attr::AttrId
- GetBoundingBox() : Util::QuadTree< TYPE >::Node, CoreGraphics::PrimitiveGroup, Graphics::Cell, Models::Model, Models::ModelNode
- GetBroadcast() : Win32::Win32Socket
- GetByteSize() : CoreGraphics::VertexComponent
- GetCameraEntity() : Graphics::View, Lighting::PSSMUtil
- GetCameraFocusEntity() : BaseGameFeature::FocusManager
- GetCameraTransform() : RenderUtil::MayaCameraUtil
- GetCastShadows() : Lighting::AbstractLightEntity
- GetCategory() : Game::Entity
- GetCategoryByIndex() : BaseGameFeature::CategoryManager
- GetCategoryByName() : BaseGameFeature::CategoryManager
- GetCell() : Graphics::GraphicsEntity
- GetChar() : CoreGraphics::DisplayEvent, Input::InputEvent
- GetCharacter() : Models::CharacterNode, Models::CharacterNodeInstance
- GetCharacter3Set() : Graphics::ActorEntity
- GetCharacterNode() : Graphics::ActorEntity
- GetCharacterPointer() : Graphics::ActorEntity
- GetCharacterSet() : Models::CharacterNode, Models::CharacterNodeInstance
- GetCharInput() : Base::KeyboardBase
- GetChildCells() : Graphics::Cell
- GetChildren() : Models::ModelNode, Models::ModelNodeInstance
- GetClassFourCC() : Core::RefCounted
- GetClassName() : Core::RefCounted
- GetClassRtti() : Core::Factory
- GetClearColor() : Base::RenderTargetBase, Frame::FramePass
- GetClearDepth() : Base::RenderTargetBase, Frame::FramePass
- GetClearStencil() : Base::RenderTargetBase, Frame::FramePass
- GetClientAddress() : Net::TcpClientConnection
- GetClientCount() : Resources::ManagedResource
- GetClipAt() : Models::CharacterNode
- GetClipDuration() : Models::CharacterNode
- GetClipIndexByName() : Models::CharacterNode
- GetCmdLineArgs() : App::Application
- GetCmdName() : Util::CmdLineArgs
- GetColor() : Lighting::AbstractLightEntity
- GetColorBufferFormat() : Base::RenderTargetBase
- GetColumnAccessMode() : Attr::AttributeTable
- GetColumnFourCC() : Attr::AttributeTable
- GetColumnId() : Attr::AttributeTable
- GetColumnIndex() : Attr::AttributeTable
- GetColumnName() : Attr::AttributeTable
- GetColumnType() : Attr::AttributeTable
- GetCommandByIndex() : Scripting::ScriptServer
- GetCommandByName() : Scripting::ScriptServer
- GetCompanyName() : App::Application, Core::CoreServer
- GetComponentAt() : Base::VertexLayoutBase
- GetComponentByteOffset() : Base::VertexLayoutBase
- GetContent() : IO::XmlReader
- GetCoreId() : Win32::Win32Thread
- GetCurrentAdapterDisplayMode() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice
- GetCurrentNodeLineNumber() : IO::XmlReader
- GetCurrentNodeName() : IO::XmlReader
- GetCurrentNodePath() : IO::XmlReader
- GetCurrentState() : App::GameApplication
- GetCurrentStateHandler() : App::GameApplication
- GetCurve() : Models::ParticleSystemNode
- GetD3D9BaseTexture() : Direct3D9::D3D9Texture
- GetD3D9CubeTexture() : Direct3D9::D3D9Texture
- GetD3D9Effect() : Direct3D9::D3D9Shader, Direct3D9::D3D9ShaderInstance, Direct3D9::D3D9ShaderVariation
- GetD3D9EffectPool() : Direct3D9::D3D9ShaderServer
- GetD3D9IndexBuffer() : Direct3D9::D3D9IndexBuffer
- GetD3D9Technique() : Direct3D9::D3D9ShaderVariation
- GetD3D9Texture() : Direct3D9::D3D9Texture
- GetD3D9VertexBuffer() : Direct3D9::D3D9VertexBuffer
- GetD3D9VertexDeclaration() : Direct3D9::D3D9VertexLayout
- GetD3D9VolumeTexture() : Direct3D9::D3D9Texture
- GetDatabasePath() : BaseGameFeature::UserProfile
- GetDay() : Base::CalendarTimeBase
- GetDebugTextEnabled() : BaseGameFeature::LoaderServer
- GetDefaultGamePad() : Base::InputServerBase
- GetDefaultKeyboard() : Base::InputServerBase
- GetDefaultMouse() : Base::InputServerBase
- GetDefaultRenderTarget() : Base::RenderDeviceBase
- GetDefaultValue() : Attr::AttributeDefinitionBase
- GetDefaultView() : Graphics::GraphicsServer
- GetDefaultViewProxy() : AsyncGraphics::GraphicsServerProxy
- GetDeletedRowIndices() : Attr::AttributeTable
- GetDependencies() : Graphics::View
- GetDepth() : Base::TextureBase
- GetDesc() : Http::HttpRequestHandler
- GetDescription() : CoreGraphics::AdapterInfo
- GetDeviceId() : CoreGraphics::AdapterInfo
- GetDeviceName() : CoreGraphics::AdapterInfo
- GetDirect3D() : Direct3D9::D3D9RenderDevice
- GetDirect3DDevice() : Direct3D9::D3D9RenderDevice
- GetDirEntries() : IO::ZipDirEntry
- GetDisplayMode() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- GetDistance() : BaseGameFeature::MoveFollow
- GetDontLinger() : Win32::Win32Socket
- GetDriverName() : CoreGraphics::AdapterInfo
- GetDriverVersionHighPart() : CoreGraphics::AdapterInfo
- GetDriverVersionLowPart() : CoreGraphics::AdapterInfo
- GetEmissionDuration() : Models::ParticleSystemNode
- GetEntities() : Graphics::Cell, Graphics::Stage, BaseGameFeature::EntityManager, GraphicsFeature::GetGraphicsEntities
- GetEntitiesByAttr() : BaseGameFeature::EntityManager
- GetEntitiesByAttrs() : BaseGameFeature::EntityManager
- GetEntitiesByType() : Graphics::Cell, Graphics::Stage
- GetEntitiesInActivityBubble() : BaseGameFeature::EntityManager
- GetEntitiesInBox() : BaseGameFeature::EnvQueryManager
- GetEntitiesInSphere() : BaseGameFeature::EnvQueryManager
- GetEntity() : Game::Property
- GetEntityByAttr() : BaseGameFeature::EntityManager
- GetEntityByAttrs() : BaseGameFeature::EntityManager
- GetEntityByUniqueId() : BaseGameFeature::EntityManager
- GetEntityUnderMouse() : BaseGameFeature::EnvQueryManager
- GetError() : Scripting::Command, Scripting::ScriptServer
- GetErrorCode() : Win32::Win32Socket
- GetErrorString() : Win32::Win32Socket
- GetEventCode() : CoreGraphics::DisplayEvent, CoreGraphics::RenderEvent
- GetFarHeight() : AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
- GetFarWidth() : AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
- GetFeatureBits() : Base::ShaderServerBase
- GetFeatureMask() : Base::ShaderVariationBase
- GetFileEntries() : IO::ZipDirEntry
- GetFileExtension() : Util::String, Models::BinaryModelWriter, Models::ModelWriter, Models::XmlModelWriter
- GetFileSize() : Win32::Win32FSWrapper, IO::ZipFileEntry
- GetFileWriteTime() : IO::IoServer, Win32::Win32FSWrapper
- GetFloat() : Attr::Attribute, Attr::AttributeContainer,
Attr::AttributeTable, IO::XmlReader, Util::CmdLineArgs, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity

- GetFloat4(): Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlReader, Util::CmdLineArgs, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- GetFloat4Array(): Util::Variant
- GetFloat4AtIndex(): Util::CmdLineArgs
- GetFloat4DefValue(): Attr::AttrId
- GetFloatArray(): Util::Variant
- GetFloatAtIndex(): Util::CmdLineArgs
- GetFloatDefValue(): Attr::AttrId
- GetFocusEntity(): BaseGameFeature::FocusManager
- GetFormat(): Base::StreamTextureSaverBase, CoreGraphics::VertexComponent
- GetFourCC(): Attr::Attribute, Attr::AttributeDefinitionBase, Attr::AttrId, Core::Rtti
- GetFov(): AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
- GetFragGroupIndex(): Models::SkinShapeNode
- GetFragmentArray(): Models::SkinShapeNode
- GetFrameCount(): Graphics::GraphicsServer
- GetFramePassByIndex(): Frame::FrameShader
- GetFramePassByName(): Frame::FrameShader
- GetFrameShader(): Graphics::View
- GetFrameShaderByName(): Frame::FrameServer
- GetFrameShaderName(): AsyncGraphics::ViewProxy
- GetFrameTime(): App::AsyncRenderApplication, App::RenderApplication
- GetFromURI(): Interface::CopyFile
- GetGlobalBoundingBox(): Graphics::GraphicsEntity
- GetGlobalLightShadowBufferTexture(): Lighting::SM30ShadowServer
- GetGraphicsEntities(): BaseGameFeature::EnvEntityManager
- GetGravity(): Models::ParticleSystemNode
- GetGuid(): Attr::Attribute, Attr::AttributeContainer,
Attr::AttributeTable, Util::Variant, CoreGraphics::AdapterInfo, Models::Model, Models::ModelNode, BaseGameFeature::GlobalAttrsManager, Game::Entity

- GetGuidArray(): Util::Variant
- GetGuidDefValue(): Attr::AttrId
- GetHandlerAtIndex(): Messaging::Port
- GetHandlerByIndex(): IO::Console
- GetHeight(): Base::RenderTargetBase, Base::TextureBase, CoreGraphics::DisplayMode
- GetHelp(): Scripting::Command
- GetHighFrequencyVibrator(): Base::GamePadBase
- GetHomeDirectory(): Win32::Win32FSWrapper
- GetHorizontalRotation(): GraphicsFeature::CameraOrbit
- GetHostAddr(): Win32::Win32IpAddress
- GetHostName(): Win32::Win32IpAddress
- GetHomeDirectory(): Win32::Win32FsWrapper
- GetHostByName(): Win32::Win32IpAddress
- GetHostByName(): Win32::Win32IpAddress
- GetHostByName(): Win32::Win32IpAddress
- GetHour(): Base::CalendarTimeBase
- GetHttpMethod(): Http::HttpRequestReader
- GetHttpMethod(): Http::HttpRequestReader
- GetHwnd(): Win32::Win32DisplayDevice
- GetIconName(): AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- GetIndexBuffer(): Base::MeshBase, Base::RenderDeviceBase
- GetIndexBufferDepth(): CoreGraphics::CPUIndexBuffer
- GetIndexType(): Base::IndexBufferBase
- GetInput(): IO::Console, IO::ConsoleHandler, Win32::Win32ConsoleHandler
- GetInputFocusEntity(): BaseGameFeature::FocusManager
- GetInstanceDataset(): BaseGameFeature::CategoryManager::Category
- GetInstanceEntity(): BaseGameFeature::CategoryManager
- GetInstances(): Models::Model
- GetInstancesByAttr(): BaseGameFeature::CategoryManager
- GetInstancesByAttrs(): BaseGameFeature::CategoryManager
- GetInstanceTable(): BaseGameFeature::CategoryManager
- GetInstanceTableName(): BaseGameFeature::CategoryManager::Category
- GetInt(): Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlReader, Util::CmdLineArgs,
Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- GetIntArray() : Util::Variant
- GetIntAtIndex() : Util::CmdLineArgs
- GetIntDefValue() : Attr::AttrId
- GetInvLightProjTransform() : Lighting::AbstractLightEntity
- GetInvModelTransform() : Base::TransformDeviceBase
- GetInvModelViewTransform() : Base::TransformDeviceBase
- GetInvProjTransform() : Base::TransformDeviceBase
- GetInvTransform() : Lighting::AbstractLightEntity
- GetInvViewTransform() : Base::TransformDeviceBase
- GetJoint() : Graphics::ActorEntity, Models::CharacterNode
- GetJointIndex() : Models::SkinShapeNode
- GetJointIndexByName() : Graphics::ActorEntity
- GetJointMatrixByIndex() : Graphics::ActorEntity
- GetJointMatrixByName() : Graphics::ActorEntity
- GetJointPaletteSize() : Models::SkinShapeNode
- GetKeepAlive() : Win32::Win32Socket
- GetKey() : CoreGraphics::DisplayEvent, Input::InputEvent
- GetKeyboardCaptureHandler() : Base::InputServerBase
- GetLastError() : Win32::SysFunc
- GetLevelName() : App::GameStateHandler
- GetLightDir() : Lighting::PSSMUtil
- GetLightDirection() : Lighting::GlobalLightEntity
- GetLightingMode() : Frame::FrameBatch
- GetLightType() : Lighting::AbstractLightEntity
- GetLinks() : Graphics::GraphicsEntity
- GetLoader() : Resources::Resource
- GetLocalBoundingBox() : Graphics::GraphicsEntity
- GetLocalJointMatrix() : Graphics::ActorEntity
- GetLocalLightShadowBufferTexture() : Lighting::SM30ShadowServer
- GetLocalTime() : Base::CalendarTimeBase, Win32::Win32CalendarTime
- GetLocalTransform() : Models::TransformNodeInstance
- GetLoop() : Models::ParticleSystemNode
- GetLowFrequencyVibrator() : Base::GamePadBase
- GetMainRenderTarget() : Frame::FrameShader
- GetManagedAnimation() : Models::CharacterNode
- GetManagedMesh() : Models::ShapeNode
- GetMapperByResourceType() : Resources::ResourceManager
- GetMaterialUnderMouse() : BaseGameFeature::EnvQueryManager
- getmatrix() : Math::transform44
- GetMatrix44() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlReader, Util::CmdLineArgs, Util::Variant, Models::Model, Models::ModelNode
- GetMatrix44Array() : Util::Variant
- GetMatrix44AtIndex() : Util::CmdLineArgs
- GetMatrix44DefValue() : Attr::AttrId
- GetMaxMsgSize() : Win32::Win32Socket
- GetMaxNumPlayers() : Base::GamePadBase
- GetMaxProgressValue() : BaseGameFeature::LoaderServer
- GetMaxScreenSpaceSize() : Resources::ManagedResource
- GetMaxTriggerDistance() : BaseGameFeature::EntityManager
- GetMediaType() : IO::Stream
- GetMesh() : Resources::ManagedMesh
- GetMethod() : Http::HttpRequest
- GetMilliSecond() : Base::CalendarTimeBase
- GetMinute() : Base::CalendarTimeBase
- GetMipLevel() : Base::StreamTextureSaverBase
- GetModel() : Models::ManagedModel, Models::ModelInstance, Models::ModelNode
- GetModelEntity() : Models::ModelInstance
- GetModelInstance() : Graphics::ModelEntity, Models::ModelNodeInstance
- GetModelNode() : Models::ModelNodeInstance
- GetModelResId() : Models::ModelReader
- GetModelResourceMapper() : Models::ModelServer
- GetModelResourceState() : Graphics::ModelEntity
- GetModelTransform() : Base::TransformDeviceBase, Models::TransformNodeInstance
- GetModelViewProjTransform() : Base::TransformDeviceBase
- GetModelViewTransform() : Base::TransformDeviceBase
- GetModifiedRowsExcludeNewAndDeletedRows() : Attr::AttributeTable
- GetModifiedTracking() : \texttt{Attr::AttributeTable}
- GetMonth() : \texttt{Base::CalendarTimeBase}
- GetMountedZipArchives() : \texttt{IO::ZipFileSystem}
- GetMouseButton() : \texttt{Input::InputEvent, CoreGraphics::DisplayEvent}
- GetMouseCaptureHandler() : \texttt{Base::InputServerBase}
- GetMouseExcludeSet() : \texttt{BaseGameFeature::EnvQueryManager}
- GetMouseMovement() : \texttt{Win32::Win32InputServer}
- GetMousePos3d() : \texttt{BaseGameFeature::EnvQueryManager}
- GetMovement() : \texttt{Base::MouseBase, Win32::Win32Mouse}
- GetMyThreadName() : \texttt{Win32::Win32Thread}
- GetName() : \texttt{BaseGameFeature::CategoryManager::Category, Base::ShaderVariationBase, Scripting::Command, Http::HttpRequestHandler, Attr::Attribute, Attr::AttributeDefinitionBase, Attr::AttrId, Core::Rtti, IO::Assign, Win32::Win32Heap, Messaging::AsyncPort, Win32::Win32Thread, AsyncGraphics::ViewProxy, Base::ShaderVariableBase, Frame::FramePass, Frame::FramePostEffect, Frame::FrameShader, Graphics::Stage, Graphics::View, Models::ModelInstance, BaseGameFeature::UserProfile}
- GetNearHeight() : \texttt{AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity}
- GetNearWidth() : \texttt{Graphics::CameraEntity, AsyncGraphics::CameraEntityProxy}
- GetNewColumnIndices() : \texttt{Attr::AttributeTable}
- GetNewRowIndices() : \texttt{Attr::AttributeTable}
- GetNodeFilter() : \texttt{Frame::FrameBatch}
- GetNodeInstances() : \texttt{Models::ModelInstance}
- GetNoDelay() : \texttt{Win32::Win32Socket}
- GetNodes() : \texttt{Models::Model}
- GetNormMousePos() : \texttt{CoreGraphics::DisplayEvent, Input::InputEvent}
- GetNumArgs() : \texttt{Scripting::ArgsBlock, Util::CmdLineArgs}
- GetNumArrayElements() : \texttt{Base::ShaderVariableBase}
- GetNumBatches() : \texttt{Frame::FramePass}
- GetNumberOfIndices() : \texttt{Resources::DynamicMeshResourceLoader}
- GetNumberOfVertices() : Resources::DynamicMeshResourceLoader
- GetNumCategories() : BaseGameFeature::CategoryManager
- GetNumClips() : Models::CharacterNode
- GetNumColorBuffers() : Base::RenderTargetBase
- GetNumComponents() : Base::VertexLayoutBase
- GetNumEntitiesInHierarchy() : Graphics::Cell
- GetNumEntitiesInHierarchyByType() : Graphics::Cell
- GetNumEntitiesInHierarchyByTypeMask() : Graphics::Cell
- GetNumFragments() : Models::SkinShapeNode
- GetNumFramePasses() : Frame::FrameShader
- GetNumHandlers() : IO::Console, Messaging::Port
- GetNumIndices() : CoreGraphics::PrimitiveGroup, Base::IndexBufferBase
- GetNumInstances() : BaseGameFeature::CategoryManager
- GetNumJoints() : Models::CharacterNode
- GetNumMipLevels() : Base::TextureBase
- GetNumPasses() : Base::ShaderVariationBase
- GetNumPostEffects() : Frame::FrameShader
- GetNumPrimitiveGroups() : Base::MeshBase
- GetNumPrimitives() : CoreGraphics::PrimitiveGroup
- GetNumRenderTargets() : Frame::FrameShader
- GetNumRows() : Attr::AttributeTable
- GetNumSharedVariables() : Base::ShaderServerBase, Direct3D9::D3D9ShaderServer
- GetNumSkins() : Models::CharacterNode
- GetNumStates() : App::GameApplication
- GetNumVariables() : Base::ShaderInstanceBase, Frame::FramePass, Frame::FramePostEffect, Frame::FrameBatch
- GetNumVariations() : Base::ShaderInstanceBase, Models::CharacterNode
- GetNumVertices() : Base::VertexBufferBase, CoreGraphics::PrimitiveGroup
- GetObject() : Util::Proxy< TYPE >, Util::Variant
- GetObjectRefCount() : Util::Proxy< TYPE >
- GetObtainFocus() : GraphicsFeature::CameraFocus,
GraphicsFeature::InputFocus
- GetOptBool() : IO::XmlReader
- GetOptFloat() : IO::XmlReader
- GetOptFloat4() : IO::XmlReader
- GetOptInt() : IO::XmlReader
- GetOptMatrix44() : IO::XmlReader
- GetOptString() : IO::XmlReader
- GetOriginalShader() : Base::ShaderInstanceBase
- GetOverlayAnimation() : Graphics::ActorEntity
- GetOverlayAnimationDuration() : Graphics::ActorEntity
- GetOverlayClip() : Graphics::ActorEntity
- GetParent() : Core::Rtti, Models::ModelNode, Models::ModelNodeInstance
- GetParentCell() : Graphics::Cell
- GetParticleMesh() : Models::ParticleSystemNode
- GetPassword() : IO::ZipArchive
- GetPath() : IO::Assign
- GetPhysicsEntity() : PhysicsFeature::PhysicsProperty
- GetPixelFormat() : Base::TextureBase, CoreGraphics::DisplayMode
- GetPixelPosition() : Base::MouseMove
- GetPlaceholderResourceId() : Resources::ResourceMapper
- GetPlayerIndex() : Base::GamePadBase
- GetPort() : Win32::Win32IpAddress, Http::HttpServer
- GetPosition() : IO::MemoryStream, IO::Stream
- getposition() : Math::transform44
- GetPosition() : IO::FileStream, IO::ZipFileStream, Models::TransformNode, Models::TransformNodeInstance
- GetPostEffectByIndex() : Frame::FrameShader
- GetPostEffectByName() : Frame::FrameShader
- GetPreShaders() : Base::ShaderInstanceBase
- GetPrimitiveGroup() : Base::RenderDeviceBase
- GetPrimitiveGroupAtIndex() : Base::MeshBase
- GetPrimitiveTopology() : CoreGraphics::PrimitiveGroup
- GetPriority() : Resources::ManagedResource, Win32::Win32Thread
- GetProfileDirectory() : BaseGameFeature::UserProfile
- GetProfileRootDirectory() : BaseGameFeature::UserProfile
- GetProgressResource() : BaseGameFeature::LoaderServer
- GetProgressText() : BaseGameFeature::LoaderServer
- GetProjMapUvOffsetAndScale() : Lighting::AbstractLightEntity
- GetProjTransform() : AsyncGraphics::CameraEntityProxy, Base::TransformDeviceBase, Graphics::CameraEntity, Lighting::AbstractLightEntity
- GetPSSMSplitDistances() : Lighting::SM30ShadowServer
- GetPSSMSplitLightProjTransforms() : Lighting::SM30ShadowServer
- GetPtr() : Util::Blob
- GetReadWriteColumnIndices() : Attr::AttributeTable
- GetRecvBufSize() : Win32::Win32Socket
- GetRecvStream() : Net::TcpClientConnection, Net::TcpClient
- GetRefCount() : Core::RefCounted
- GetRelativeDistanceChange() : GraphicsFeature::CameraDistance
- GetRenderCount() : Resources::ManagedResource
- GetRenderOldestFirst() : Models::ParticleSystemNode
- GetRenderTarget() : Frame::FramePostEffect, Graphics::View, Frame::FramePass
- GetRenderTargetByIndex() : Frame::FrameShader
- GetRenderTargetByName() : Frame::FrameShader
- GetRequestURI() : Http::HttpRequestReader
- GetResolveRect() : Base::RenderTargetBase
- GetResolveTexture() : Base::RenderTargetBase
- GetResolveTextureHeight() : Base::RenderTargetBase
- GetResolveTextureResourceId() : Base::RenderTargetBase
- GetResource() : Resources::ManagedResource, Resources::ResourceSaver, Resources::ResourceLoader
- GetResourceId() : Resources::Resource, Graphics::ModelEntity, Resources::ManagedResource
- GetResourceState() : Models::CharacterNode, Models::Model, Models::ModelNode, Models::StateNode, Models::ShapeNode
- GetResourceType() : Resources::SimpleResourceMapper, Resources::ManagedResource, Resources::ResourceMapper
- GetResponseContentStream() : Http::HttpRequest
- GetResult() : Interface::IOMessage, Util::Crc,
Interface::CopyFile
- GetResults() : Scripting::Command
- GetReturnCode() : App::Application
- GetReUseAddr() : Win32::Win32Socket
- GetRevision() : CoreGraphics::AdapterInfo
- GetRGBCurve() : Models::ParticleSystemNode
- GetRootCell() : Graphics::Stage
- GetRootLocation() : Http::HttpRequestHandler
- GetRotate() : Models::TransformNodeInstance, Models::TransformNode
- getrotate() : Math::transform44
- GetRotatePivot() : Models::TransformNodeInstance
- getrotatepivot() : Math::transform44
- GetRotatePivot() : Models::TransformNode
- GetRowUserData() : Attr::AttributeTable
- GetSaveGame() : App::GameStateHandler
- GetSaveGameDirectory() : BaseGameFeature::UserProfile
- GetSaveGamePath() : BaseGameFeature::UserProfile
- GetSaver() : Resources::Resource
- GetScale() : Models::TransformNodeInstance
- getscale() : Math::transform44
- GetScale() : Models::TransformNode
- GetScalePivot() : Models::TransformNodeInstance
- getscalepivot() : Math::transform44
- GetScalePivot() : Models::TransformNode
- GetScreenPosition() : Base::MouseBase
- GetSecond() : Base::CalendarTimeBase
- GetSemantic() : Base::ShaderVariableBase
- GetSemanticIndex() : CoreGraphics::VertexComponent
- GetSemanticName() : CoreGraphics::VertexComponent
- GetSendBufSize() : Win32::Win32Socket
- GetSendStream() : Net::TcpClient, Net::TcpClientConnection
- GetServerAddress() : Net::TcpClient
- GetSetupMode() : App::GameStateHandler
- GetShader() : Frame::FramePostEffect, Frame::FrameBatch, Frame::FramePass
- GetShaderFeatures() : Frame::FrameBatch
- GetShaderInstance() : Models::StateNode
- GetShaderParamBindMode() : Base::ShaderServerBase
- GetShaderVariable() : `Base::ShaderVariableInstanceBase`
- GetShaderVariableInstance() : `Models::StateNodeInstance`
- GetShadowBufferUvOffsetAndScale() : `Lighting::AbstractLightEntity`
- GetSharedResources() : `Resources::SharedResourceServer`
- GetSharedResourcesByType() : `Resources::SharedResourceServer`
- GetSharedVariableByIndex() : `Direct3D9::D3D9ShaderServer`, `Base::ShaderServerBase`
- GetSharedVariableByName() : `Direct3D9::D3D9ShaderServer`, `Base::ShaderServerBase`
- GetSharedVariableBySemantic() : `Direct3D9::D3D9ShaderServer`, `Base::ShaderServerBase`
- GetSignature() : `CoreGraphics::VertexComponent`
- GetSize() : `IO::ZipFileStream`, `IO::MemoryStream`, `IO::Stream`, `IO::FileStream`
- GetSkinIndexByName() : `Models::CharacterNode`
- GetSkinInfoArray() : `Models::CharacterNode`
- GetSkinInfoAt() : `Models::CharacterNode`
- GetSkinList() : `Graphics::ActorEntity`
- GetSockAddr() : `Win32::Win32IpAddress`
- GetSortingMode() : `Frame::FrameBatch`
- GetSplitDistances() : `Lighting::PSSMUtil`
- GetSplitLightProjTransform() : `Lighting::PSSMUtil`
- GetSplitLightProjTransforms() : `Lighting::PSSMUtil`
- GetSplitLightTransform() : `Lighting::PSSMUtil`
- GetSplitProjTransform() : `Lighting::PSSMUtil`
- GetStackSize() : `Win32::Win32Thread`
- GetStage() : `Graphics::Cell`, `Graphics::View`, `Graphics::GraphicsEntity`
- GetStageBuilder() : `Graphics::Stage`
- GetStageByName() : `Graphics::GraphicsServer`
- GetStageName() : `AsyncGraphics::ViewProxy`
- GetStageProxies() : `AsyncGraphics::GraphicsServerProxy`
- GetStageProxy() : `AsyncGraphics::GraphicsEntityProxy`
- GetStageProxyByName() : `AsyncGraphics::GraphicsServerProxy`
- GetStages() : `Graphics::GraphicsServer`
- GetState() : `Resources::ManagedResource`, `Resources::SharedResourceServer`
Resources::Resource, Resources::ResourceLoader
- GetStateHandlerAt() : App::GameApplication
- GetStatus() : Http::HttpRequest
- GetStream() : IO::StreamReader, Interface::ReadStream, IO::StreamWriter, Base::StreamTextureSaverBase, Interface::WriteStream
- GetStreamByteOrder() : IO::BinaryWriter, IO::BinaryReader
- GetStreamClassByUriScheme() : IO::IoServer
- GetString() : Attr::AttributeTable, Util::CmdLineArgs, Attr::AttributeContainer, Util::Variant, Models::Model, Game::Entity, IO::XmlReader, BaseGameFeature::UserProfile, Models::ModelNode, Attr::AttributeContainer, BaseGameFeature::GlobalAttrsManager, Attr::AttributeTable, Attr::Attribute
- GetStringArray() : Util::Variant
- GetStringAtIndex() : Util::CmdLineArgs
- GetStringDefValue() : Attr::AttrId
- GetSubSystemId() : CoreGraphics::AdapterInfo
- GetSubType() : IO::MediaType
- GetSyntax() : Scripting::Command
- GetSystemTime() : Win32::Win32CalendarTime, Base::CalendarTimeBase
- GetTail() : IO::URI
- GetTargetEntityId() : BaseGameFeature::MoveFollow
- GetTempDirectory() : Win32::Win32FSWrapper
- GetTemplateDataset() :
  BaseGameFeature::CategoryManager::Category
- GetTemplateTable() : BaseGameFeature::CategoryManager
- GetTemplateName() :
  BaseGameFeature::CategoryManager::Category
- GetTexture() : Resources::ManagedTexture
- GetThreadPriority() : Messaging::AsyncPort
- GetThreadStackSize() : Messaging::AsyncPort
- GetTicks() : Win32::Win32Timer
- GetTime() : Graphics::GraphicsEntity, Win32::Win32Timer, Models::ModelInstance, App::RenderApplication, App::AsyncRenderApplication
- GetToURI() : Interface::CopyFile
GetTransform() : Models::ModellInstance, AsyncGraphics::GraphicsEntityProxy, Graphics::GraphicsEntity

GetType() : Base::ShaderVariableBase, Base::TextureBase, Models::ModelNode, Graphics::GraphicsEntity, Frame::FrameBatch, AsyncGraphics::GraphicsEntityProxy, IO::MediaType, Input::InputEvent, Util::Variant

GetUniqueId() : Game::Entity

GetUpVector() : BaseGameFeature::EnvQueryManager

GetURI() : IO::ZipArchive, IO::Stream, Interface::IOMessage, Http::HttpRequest

GetUsage() : Base::ResourceBase

GetUseCount() : Resources::Resource

GetUserDirectory() : Win32::Win32FSWrapper

GetUserProfile() : BaseGameFeature::LoaderServer

GetValue() : Attr::Attribute

GetValueType() : Attr::AttrId, Attr::Attribute, Attr::AttributeDefinitionBase

GetVariableByIndex() : Frame::FramePostEffect, Frame::FrameBatch, Frame::FramePass, Base::ShaderInstanceBase

GetVariableByName() : Base::ShaderInstanceBase

GetVariableBySemantic() : Base::ShaderInstanceBase

GetVariationByFeatureMask() : Base::ShaderInstanceBase

GetVariationByIndex() : Base::ShaderInstanceBase

GetVariationIndexByName() : Models::CharacterNode

GetVariationJointsAt() : Models::CharacterNode

GetVariationNameAt() : Models::CharacterNode

GetVariationsUri() : Models::CharacterNode

GetVendorId() : CoreGraphics::AdapterInfo

GetVertexBuffer() : Base::RenderDeviceBase, Base::MeshBase

GetVertexByteSize() : Base::VertexLayoutBase

GetVertexLayout() : Base::VertexBufferBase

GetVerticalRotation() : GraphicsFeature::CameraOrbit

GetViewByName() : Graphics::GraphicsServer

GetViewClass() : AsyncGraphics::ViewProxy

GetViewName() : AsyncGraphics::CameraEntityProxy

GetViewProjTransform() : Graphics::CameraEntity,
AsyncGraphics::CameraEntityProxy,
Base::TransformDeviceBase
- getViewProxies() : AsyncGraphics::GraphicsServerProxy
- getViewProxyByName() : AsyncGraphics::GraphicsServerProxy
- getViews() : Graphics::GraphicsServer
- getViewTransform() : Graphics::CameraEntity,
AsyncGraphics::CameraEntityProxy,
Base::TransformDeviceBase
- getVisibleModelNodeInstances() : Models::VisResolver,
Models::ModelNode
- getVisibleModelNodes() : Models::VisResolver
- getVisibleModels() : Models::VisResolver
- getWaitForMessages() : Messaging::AsyncPort
- getWeekday() : Base::CalendarTimeBase
- getWidth() : Base::TextureBase, CoreGraphics::DisplayMode,
Base::RenderTargetBase
- getWindowTitle() : Base::DisplayDeviceBase
- getXPos() : CoreGraphics::DisplayMode
- getYear() : Base::CalendarTimeBase
- getYPos() : CoreGraphics::DisplayMode
- getZFar() : Graphics::CameraEntity,
AsyncGraphics::CameraEntityProxy
- getZNear() : Graphics::CameraEntity,
AsyncGraphics::CameraEntityProxy
- globalAttrsManager() :
  BaseGameFeature::GlobalAttrsManager
- globalLightEntity() : Lighting::GlobalLightEntity
- graphicsEntity() : Graphics::GraphicsEntity
- graphicsEntityProxy() : AsyncGraphics::GraphicsEntityProxy
- graphicsServer() : Graphics::GraphicsServer
- graphicsServerProxy() :
  AsyncGraphics::GraphicsServerProxy
- gt() : Math::float2
- guidAttrId() : Attr::GuidAttrId
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **h** -

- **Handled()** : `Messaging::Message`
- **HandleDelayedJobs()** : `BaseGameFeature::EntityManager`
- **HandleEvent()** : `CoreGraphics::RenderEventHandler`, `CoreGraphics::ThreadSafeDisplayEventHandler`, `CoreGraphics::DisplayEventHandler`, `CoreGraphics::ThreadSafeRenderEventHandler`, `Win32::Win32InputDisplayEventHandler`
- **HandleLeftMouseBtnDown()** : `PhysicsFeature::MouseGripperProperty`
- **HandleLeftMouseBtnUp()** : `PhysicsFeature::MouseGripperProperty`
- **HandleMessage()** : `Messaging::Dispatcher`, `Messaging::Handler`, `Messaging::Port`, `AsyncGraphics::AsyncGraphicsHandler`, `Game::Property`, `PhysicsFeature::PhysicsProperty`
- **HandlePendingEvents()** : `CoreGraphics::ThreadSafeDisplayEventHandler`, `CoreGraphics::ThreadSafeRenderEventHandler`
- **Handler()** : `Messaging::Handler`
- **HandleRequest()** : `Debug::CorePageHandler`, `Http::DefaultHttpRequestHandler`, `Http::HttpRequestHandler`, `Debug::IoPageHandler`, `Debug::MemoryPageHandler`, `Debug::ScriptingPageHandler`, `Debug::DisplayPageHandler`, `Debug::MeshPageHandler`, `Debug::ShaderPageHandler`, `Debug::TexturePageHandler`
- **HasActiveEntities()** : `BaseGameFeature::EntityManager`
- HasArg() : Scripting::ArgBlock, Util::CmdLineArgs
- HasAssign() : IO::IoServer
- HasAttr() : Attr::AttributeContainer, IO::XmlReader, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- HasCategory() : BaseGameFeature::CategoryManager
- HasCharacter3Set() : Graphics::ActorEntity
- HasColorBuffer() : Base::RenderTargetBase
- HasColumn() : Attr::AttributeTable
- HasCommand() : Scripting::ScriptServer
- HasComponent() : Base::VertexLayoutBase
- HasContent() : IO::XmlReader
- HasDeletedRows() : Attr::AttributeTable
- HasDepthStencilBuffer() : Base::RenderTargetBase
- HasEntity() : Game::Entity
- HasFramePass() : Frame::FrameShader
- HasFrameShader() : Frame::FrameServer
- HashCode() : Util::Blob, Util::String, Win32::Win32Guid
- HashTable() : Util::HashTable<KEYTYPE, VALUETYPE>
- HasIndexBuffer() : Base::MeshBase
- HasInput() : IO::Console, IO::ConsoleHandler, Win32::Win32ConsoleHandler
- HasInstanceDataset() :
  BaseGameFeature::CategoryManager::Category
- HasInstanceTable() : BaseGameFeature::CategoryManager
- HasManagedModel() : Models::ModelServer
- HasManagedResource() : Resources::ResourceManager
- HasMapper() : Resources::ResourceManager
- HasModifiedRows() : Attr::AttributeTable
- HasMouseIntersection() :
  BaseGameFeature::EnvQueryManager
- HasNewRows() : Attr::AttributeTable
- HasNode() : IO::XmlReader, Models::Model
- HasNodeInstance() : Models::ModelInstance
- HasParent() : Util::SimpleTree<VALUETYPE>::Node, Models::ModelNode, Models::ModelNodeInstance
- HasPort() : Messaging::Dispatcher
- HasPostEffect() : Frame::FrameShader
- HasRecvData() : \texttt{Win32::Win32Socket}
- HasRenderTarget() : \texttt{Frame::FrameShader}
- HasResolveTexture() : \texttt{Base::RenderTargetBase}
- HasShader() : \texttt{Base::ShaderServerBase}
- HasShaderVariableInstance() : \texttt{Models::StateNodeInstance}
- HasSharedResource() : \texttt{Resources::SharedResourceServer}
- HasSharedVariableByName() : \texttt{Base::ShaderServerBase}, \texttt{Direct3D9::D3D9ShaderServer}
- HasSharedVariableBySemantic() : \texttt{Base::ShaderServerBase}, \texttt{Direct3D9::D3D9ShaderServer}
- HasStage() : \texttt{Graphics::GraphicsServer}
- HasStageProxy() : \texttt{AsyncGraphics::GraphicsServerProxy}
- HasStarted() : \texttt{Game::GameServer}
- HasStream() : \texttt{IO::StreamReader}, \texttt{IO::StreamWriter}
- HasTemplateDataset() :
  \texttt{BaseGameFeature::CategoryManager::Category}
- HasTemplateTable() : \texttt{BaseGameFeature::CategoryManager}
- HasVariableByName() : \texttt{Base::ShaderInstanceBase}
- HasVariableBySemantic() : \texttt{Base::ShaderInstanceBase}
- HasVariation() : \texttt{Base::ShaderInstanceBase}
- HasVertexBuffer() : \texttt{Base::MeshBase}
- HasView() : \texttt{Graphics::GraphicsServer}
- HasViewProxy() : \texttt{AsyncGraphics::GraphicsServerProxy}
- height() : \texttt{Math::rectangle<TYPE>}
- Height() : \texttt{Util::FixedTable<TYPE>}
- HorizontalRule() : \texttt{Http::HtmlPageWriter}
- Host() : \texttt{IO::URI}
- HtmlPageWriter() : \texttt{Http::HtmlPageWriter}
- HttpRequest() : \texttt{Http::HttpRequest}
- HttpRequestHandler() : \texttt{Http::HttpRequestHandler}
- HttpRequestReader() : \texttt{Http::HttpRequestReader}
- HttpServer() : \texttt{Http::HttpServer}
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- i -

  - Id() : `Messaging::Id`
  - IncrClientCount() : `Resources::ManagedResource`
  - Increment() : `Win32::Win32Interlocked`
  - IncrUseCount() : `Resources::Resource`
  - IndexBufferBase() : `Base::IndexBufferBase`
  - Initialize() : `Util::QuadTree< TYPE >::Node`
  - InputEvent() : `Input::InputEvent`
  - InputFocus() : `GraphicsFeature::InputFocus`
  - InputHandler() : `Input::InputHandler`
  - InputServer() : `Input::InputServer`
  - InputServerBase() : `Base::InputServerBase`
  - Insert() : `Threading::SafePriorityQueue< PRITYPE, TYPE >`, `Util::Array< TYPE >`, `Util::SimpleTree< VALUETYPE >::Node`
  - InsertEntity() : `Graphics::Cell`
  - InsertSorted() : `Util::Array< TYPE >`
  - inside() : `Math::rectangle< TYPE >`, `Math::sphere`
  - Instance() : `Core::Factory`
  - IntAttrId() : `Attr::IntAttrId`
  - intersect() : `Math::line`
  - intersect_sweep() : `Math::sphere`
  - intersects() : `Math::bbox`, `Math::sphere`
  - IOMessage() : `Interface::IOMessage`
  - IoPageHandler() : `Debug::IoPageHandler`
  - IoServer() : `IO::IoServer`
  - IsA() : `Core::RefCounted`
  - IsActive() : `Graphics::GraphicsEntity`, `Game::Entity`
Game::FeatureUnit, Game::Manager, Game::Property

- IsAlwaysOnTop() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- IsArray() : Base::ShaderVariableBase
- IsAsyncEnabled() : Resources::Resource, Resources::ResourceMapper
- IsAttached() : Input::InputHandler
- IsAttachedToCell() : Graphics::GraphicsEntity
- IsAttachedToModel() : Models::ModelInstance, Models::ModelNode
- IsAttachedToModelInstance() : Models::ModelNodeInstance
- IsAttachedToResource() : Resources::ResourceLoader, Resources::ResourceSaver
- IsAttachedToResourceManager() : Resources::ResourceMapper
- IsAttachedToServer() : Graphics::Stage, Graphics::View
- IsAttachedToStage() : Graphics::Cell, Graphics::GraphicsEntity
- IsAttachedToStageProxy() : AsyncGraphics::GraphicsEntityProxy
- IsAttachedToView() : Graphics::CameraEntity
- IsBlocking() : Net::TcpClient
- IsBound() : Win32::Win32Socket
- IsCapturing() : Input::InputHandler
- IsConnected() : Win32::Win32Socket, Base::GamePadBase, Net::TcpClient, Net::TcpClientConnection
- IsDefaultRenderTarget() : Base::RenderTargetBase
- IsDefaultView() : AsyncGraphics::ViewProxy
- IsDerivedFrom() : Core::Rtti
- IsDeviceName() : Win32::Win32FSWrapper
- isdirty() : Math::transform44
- IsDisplayModeSwitchEnabled() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- IsEmpty() : Attr::AttributeDefinitionBase, Attr::AttrId
- IsEmpty() : IO::URI, Scripting::ArgsBlock, Threading::SafePriorityQueue< PRITYPE, TYPE >, Threading::SafeQueue< TYPE >, Util::Array< TYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::List< TYPE >, Util::Queue<
TYPE > , Util::SimpleTree< VALUETYPE >::Node , Util::Stack<
  TYPE > , Util::String

- IsEnabled() : PhysicsFeature::PhysicsProperty
- IsEntityHandleValid() : AsyncGraphics::GraphicsEntityProxy
- IsFullscreen() : AsyncGraphics::DisplayProxy ,
  Base::DisplayDeviceBase
- IsInBeginFrame() : Base::RenderDeviceBase
- IsInetAddr() : Win32::Win32IpAddress
- IsInFocus() : BaseGameFeature::EntityManager
- IsInstanceOf() : Core::RefCounted
- IsLoaded() : Resources::Resource ,
  BaseGameFeature::UserProfile
- IsMapped() : IO::Stream
- IsMemoryMappingEnabled() : IO::BinaryWriter ,
  IO::BinaryReader
- IsModified() : Attr::AttributeTable
- IsMounted() : IO::ZipFileSystem
- IsOpen() : CoreGraphics::VertexLayoutServer ,
  Frame::FrameServer , Graphics::GraphicsServer ,
  Base::InputServerBase , Models::ModelServer ,
  Models::VisResolver , Resources::ResourceManager ,
  BaseGameFeature::LoaderServer , Scripting::ScriptServer ,
  Base::RenderDeviceBase , IO::Stream , Core::CoreServer ,
  Lighting::LightServerBase , App::Application ,
  Http::HttpServer , IO::Console , IO::ConsoleHandler ,
  IO::StreamReader , IO::StreamWriter , IO::ZipArchive ,
  Messaging::AsyncPort , Messaging::Handler , Net::TcpServer ,
  Win32::Win32Socket , Resources::SharedResourceServer ,
  AsyncGraphics::DisplayProxy ,
  AsyncGraphics::GraphicsServerProxy ,
  Base::DisplayDeviceBase , Base::ShaderServerBase ,
  Base::ShapeRendererBase , Base::TransformDeviceBase
- IsOrthogonal() : Graphics::CameraEntity ,
  AsyncGraphics::CameraEntityProxy
- IsOverlayAnimationActive() : Graphics::ActorEntity
- IsPending() : Resources::Resource
- IsPerspective() : AsyncGraphics::CameraEntityProxy ,
  Graphics::CameraEntity
- IsQuitRequested() : App::RenderApplication ,
Base::InputServerBase, Game::GameServer, App::AsyncRenderApplication
- IsReadOnly() : Win32::Win32FSWrapper, IO::IoServer
- IsRegistered() : Scripting::Command
- IsResolved() : Models::VisResolveContainer< TYPE >
- IsRowDeleted() : Attr::AttributeTable
- IsRowModified() : Attr::AttributeTable
- IsRunning() : Win32::Win32Thread
- IsSpecial() : BaseGameFeature::CategoryManager::Category
- IsTripleBufferingEnabled() : Base::DisplayDeviceBase, AsyncGraphics::DisplayProxy
- IsUriSchemeRegistered() : IO::IoServer
- IsValid() : Base::RenderTargetBase, Util::Atom< TYPE >, Win32::Win32Guid, IO::MediaType, Util::Proxy< TYPE >, IO::URI, AsyncGraphics::ViewProxy, Util::Blob, Graphics::GraphicsEntity, BaseGameFeature::CategoryManager::Entry, Util::String, Base::ShaderInstanceBase, Base::VertexLayoutBase, AsyncGraphics::GraphicsEntityProxy, Util::FourCC, Attr::AttrId
- IsValidBool() : Util::String
- IsValidFloat() : Util::String
- IsValidFloat4() : Util::String
- IsValidFourCC() : Attr::AttrId
- IsValidHttpRequest() : Http::HttpRequestReader
- IsValidInt() : Util::String
- IsValidMatrix44() : Util::String
- IsValidName() : Attr::AttrId
- IsVerticalSyncEnabled() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- IsVirtual() : BaseGameFeature::CategoryManager::Category
- IsZipArchiveMounted() : IO::IoServer
- IsZipFileSystemEnabled() : IO::IoServer
- Iterator() : Util::List< TYPE >::Iterator, Util::FixedArray< TYPE >, Util::Array< TYPE >
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
- KeyDown() : Base::KeyboardBase
- KeyPressed() : Base::KeyboardBase
- 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
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- le() : Math::float2
- Leave() : Win32::Win32CriticalSection
- length() : Math::line
- Length() : Util::String
- length() : Math::float2
- lengthsq() : Math::float2, Math::line
- Level() : Util::QuadTree< TYPE >::Node
- LevelExists() : BaseGameFeature::CategoryManager
- LightServer() : Lighting::LightServer
- LightServerBase() : Lighting::LightServerBase
- line() : Math::line
- LineBreak() : Http::HtmlPageWriter
- LinkType : Graphics::GraphicsEntity
- List() : Util::List< TYPE >
- ListDirectories() : IO::IoServer, Win32::Win32FSWrapper, IO::ZipArchive
- Listen() : Win32::Win32Socket
- ListFiles() : IO::IoServer, Win32::Win32FSWrapper, IO::ZipArchive
- Load() : Resources::Resource, BaseGameFeature::EntityLoader, BaseGameFeature::EntityLoaderBase, BaseGameFeature::EnvironmentLoader, BaseGameFeature::LevelLoader, BaseGameFeature::UserProfile
- LoadAttributes() : BaseGameFeature::GlobalAttrsManager
- `LoadEntities()` : `BaseGameFeature::LoaderServer`
- `LoaderServer()` : `BaseGameFeature::LoaderServer`
- `LoadFailed()` : `Resources::Resource`
- `LoadFrameShaders()` : `Frame::FrameShaderLoader`
- `LoadFromAttrs()` : `Models::Model`, `Models::ModelNode`, `Models::TransformNode`
- `LoadInstances()` : `BaseGameFeature::CategoryManager`
- `LoadLevel()` : `BaseGameFeature::LoaderServer`
- `LoadManagedModel()` : `Models::ModelServer`
- `LoadResources()` : `Models::ModelNode`, `Models::ParticleSystemNode`, `Models::CharacterNode`, `Models::Model`, `Models::StateNode`, `Models::SkinShapeNode`, `Models::ShapeNode`
- `LoadVariation()` : `Models::CharacterNode`
- `LoadXmlTable()` : `Attr::AttributeTable`
- `LocalPath()` : `IO::URI`
- `LocalTimeToFileTime()` : `Win32::Win32CalendarTime`, `Base::CalendarTimeBase`
- `LookupManagedModel()` : `Models::ModelServer`
- `LookupManagedResource()` : `Resources::ResourceManager`
- `LookupNode()` : `Models::Model`
- `LookupNodeInstance()` : `Models::ModelInstance`
- `LookupNodeInstanceByRTTI()` : `Models::ModelInstance`
- `LookupSharedResource()` : `Resources::SharedResourceServer`
- `lt()` : `Math::float2`
- `LuaServer()` : `Scripting::LuaServer`
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **m** -

- ManagedResource() : Resources::ManagedResource
- Manager() : Game::Manager
- Map() : IO::MemoryStream, Base::TextureBase, Base::VertexBufferBase, IO::Stream, CoreGraphics::CPUIndexBuffer, CoreGraphics::CPUVertexBuffer, IO::FileStream, IO::ZipFileStream, Direct3D9::D3D9IndexBuffer, Direct3D9::D3D9Texture, Base::IndexBufferBase, Direct3D9::D3D9VertexBuffer
- MapCubeFace() : Direct3D9::D3D9Texture, Base::TextureBase
- MapInfo() : Base::TextureBase::MapInfo
- Mask : CoreGraphics::ShaderFeature
- MatchPattern() : Util::String
- Matrix44AttrId() : Attr::Matrix44AttrId
- maximize() : Math::float2
- MaxNumColorBuffers : Base::RenderTargetBase
- MayaCameraUtil() : RenderUtil::MayaCameraUtil
- MediaTyp() : IO::MediaType
- MemoryIndexBufferLoaderBase() :
  Base::MemoryIndexBufferLoaderBase
- MemoryPageHandler() : Debug::MemoryPageHandler
- MemoryStream() : IO::MemoryStream
- MemoryVertexBufferLoaderBase() :
  Base::MemoryVertexBufferLoaderBase
- MeshBase() : Base::MeshBase
- MeshPageHandler() : `Debug::MeshPageHandler`
- Message() : `Messaging::Message`
- MessageReader() : `Messaging::MessageReader`
- MessageWriter() : `Messaging::MessageWriter`
- minimize() : `Math::float2`
- Model() : `Models::Model`
- ModelEntity() : `Graphics::ModelEntity`
- ModelInstance() : `Models::ModelInstance`
- ModelNode() : `Models::ModelNode`
- ModelNodeInstance() : `Models::ModelNodeInstance`
- ModelServer() : `Models::ModelServer`
- Month : `Base::CalendarTimeBase`
- MonthToString() : `Base::CalendarTimeBase`
- Mount() : `IO::ZipFileSystem`
- MountZipArchive() : `IO::IoServer`
- MouseButton() : `Base::MouseBase`
- MouseGripperProperty() : `PhysicsFeature::MouseGripperProperty`
- MoveFollow() : `BaseGameFeature::MoveFollow`

The Nebula Device 3 documentation generated by `doxygen` at Tue Feb 19 12:16:40 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- N2ModelReader() : Models::N2ModelReader
- Name : Base::ShaderVariableBase, CoreGraphics::ShaderFeature
- Node() : Util::QuadTree<TYPE>::Node, Util::SimpleTree<VALUETYPE>::Node
- normalize() : Math::float2
- NotifyEventHandlers() : Base::DisplayDeviceBase, Base::RenderDeviceBase
- NotifyGameLoad() : Game::GameServer
- NotifyGameSave() : Game::GameServer
- NotifyVisible() : Models::ModelInstance
- nullvec() : Math::vector
- NumSplits : Lighting::PSSMUtil
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **O** -

- ObtainKeyboardCapture() : `Base::InputServerBase`
- ObtainMouseCapture() : `Base::InputServerBase`
- OnActivate() : `Graphics::GraphicsEntity`, `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`, `Graphics::ModelEntity`, `Game::FeatureUnit`, `Game::Manager`, `Graphics::ActorEntity`, `BaseGameFeature::CategoryManager`, `Game::Property`, `PhysicsFeature::MouseGripperProperty`, `BaseGameFeature::EnvQueryManager`, `PhysicsFeature::PhysicsProperty`
- OnApply() : `PreShaders::GaussianBlur5x5FilterKernel`, `PreShaders::BoxFilterKernel`
- OnAttach() : `CoreGraphics::DisplayEventHandler`, `CoreGraphics::RenderEventHandler`, `Base::GamePadBase`, `Base::KeyboardBase`, `Base::MouseBase`, `Input::InputHandler`, `PreShaders::BoxFilterKernel`, `PreShaders::GaussianBlur5x5FilterKernel`
- OnAttachToCell() : `Graphics::GraphicsEntity`
- OnAttachToModel() : `Models::ModellInstance`, `Models::ModelNode`
- OnAttachToModelInstance() : `Models::StateNodeInstance`, `Models::TransformNodeInstance`, `Models::ModelNodeInstance`, `Models::CharacterNodeInstance`, `Models::ParticleSystemNodeInstance`, `Models::SkinShapeNodeInstance`
- **OnAttachToResource()**: 
  `Resources::DynamicMeshResourceLoader`, `Resources::ResourceLoader`, `Resources::ResourceSaver`
- **OnAttachToResourceManager()**: `Resources::SimpleResourceMapper`
- **OnAttachToServer()**: `Graphics::Stage`, `Graphics::View`
- **OnAttachToStage()**: `Graphics::Cell`, `Graphics::GraphicsEntity`
- **OnAttachToView()**: `Graphics::CameraEntity`
- **OnBeginFrame()**: `Base::KeyboardBase`, `Base::MouseBase`, `Input::InputHandler`, `XInput::XInputGamePad`, `BaseGameFeature::EntityManager`, `Game::Entity`, `Game::FeatureUnit`, `Game::Manager`, `Game::Property`, `PhysicsFeature::MouseGripperProperty`
- **OnChar()**: `Win32::Win32DisplayDevice`
- **OnCloseRequested()**: `Win32::Win32DisplayDevice`
- **OnConfigureDisplay()**: `App::AsyncRenderApplication`
- **OnConfigureDisplayDevice()**: `App::RenderApplication`
- **OnCreateHandlers()**: `AsyncHttp::AsyncHttpInterface`, `Messaging::AsyncPort`, `AsyncGraphics::AsyncGraphicsInterface`
- **OnCreateManagedResource()**: `Resources::ResourceMapper`, `Resources::SimpleResourceMapper`
- **OnDeactivate()**: `Graphics::CameraEntity`, `Graphics::GraphicsEntity`, `Graphics::ModelEntity`, `BaseGameFeature::CategoryManager`, `BaseGameFeature::EntityManager`, `BaseGameFeature::EnvEntityManager`, `BaseGameFeature::EnvQueryManager`, `Game::Entity`, `Game::FeatureUnit`, `Game::Manager`, `Game::Property`, `PhysicsFeature::MouseGripperProperty`, `PhysicsFeature::PhysicsProperty`, `Graphics::ActorEntity`
- **OnDetach()**: `PreShaders::BoxFilterKernel`, `PreShaders::GaussianBlur5x5FilterKernel`
- **OnDiscardManagedResource()**: `Resources::ResourceMapper`, `Resources::SimpleResourceMapper`
- **OnEndFrame()**: `Input::InputHandler`, `BaseGameFeature::EntityManager`, `Game::FeatureUnit`, `Game::Manager`
- **OnEvent()**: `Base::KeyboardBase`, `Base::MouseBase`,
Input::InputHandler
- OnExecute() : Scripting::Command
- OnFrame() : AsyncGraphics::GraphicsServerProxy, Graphics::GraphicsServer, Base::InputServerBase, Win32::Win32InputServer, App::GameStateHandler, BaseGameFeature::EnvQueryManager, BaseGameFeature::FocusManager, Game::FeatureUnit, Game::GameServer, Game::Manager
- OnGainActivity() : Game::Entity, Game::Property
- OnHide() : Graphics::GraphicsEntity
- OnKeyDown() : Win32::Win32DisplayDevice
- OnKeyUp() : Win32::Win32DisplayDevice
- OnKillFocus() : Win32::Win32DisplayDevice
- OnLoad() : Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property
- OnLoadCancelled() : Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, Models::StreamModelLoader, Resources::ResourceLoader
- OnLoadRequested() :
  CoreGraphics::CPUMemoryIndexBufferLoader, CoreGraphics::CPUMemoryVertexBufferLoader, Direct3D9::D3D9MemoryIndexBufferLoader, Direct3D9::D3D9MemoryVertexBufferLoader, Direct3D9::D3D9StreamShaderLoader, Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, Models::StreamModelLoader, Resources::DynamicMeshResourceLoader, Resources::ResourceLoader
- OnLoseActivity() : Game::Entity, Game::Property
- OnLostDevice() : Direct3D9::D3D9ShaderInstance
- OnMinimized() : Win32::Win32DisplayDevice
- OnMouseButton() : Win32::Win32DisplayDevice
- OnMouseMove() : Win32::Win32DisplayDevice
- OnMouseWheel() : Win32::Win32DisplayDevice
- OnMoveAfter() : Game::Entity, Game::Property, PhysicsFeature::MouseGripperProperty,
PhysicsFeature::PhysicsProperty
- OnMoveBefore() : Game::Entity, Game::Property, PhysicsFeature::MouseGripperProperty
- OnNotifyVisible() : Models::ModelNodeInstance, Models::ParticleSystemNodeInstance, Models::ShapeNodeInstance, Models::SkinShapeNodeInstance
- OnObtainCapture() : Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- OnPaint() : Win32::Win32DisplayDevice
- OnPending() : Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, Models::StreamModelLoader, Resources::ResourceLoader
- OnPrepare() : Resources::ResourceMapper, Resources::SimpleResourceMapper
- OnProcessInput() : App::AsyncRenderApplication, App::AsyncViewerApplication, App::RenderApplication, App::ViewerApplication
- OnRegister() : Scripting::Command
- OnReleaseCapture() : Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- OnRemove() : CoreGraphics::DisplayEventHandler, CoreGraphics::RenderEventHandler, Input::InputHandler
- OnRemoveFromCell() : Graphics::GraphicsEntity
- OnRemoveFromModel() : Models::ModellInstance, Models::ModelNode
- OnRemoveFromModelInstance() : Models::ModelNodeInstance, Models::CharacterNodeInstance, Models::ParticleSystemNodeInstance, Models::SkinShapeNodeInstance, Models::StateNodeInstance, Models::TransformNodeInstance
- OnRemoveFromResource() : Resources::DynamicMeshResourceLoader, Resources::ResourceLoader, Resources::ResourceSaver
- OnRemoveFromResourceManager() : Resources::ResourceMapper, Resources::SimpleResourceMapper
- OnRemoveFromServer() : Graphics::Stage, Graphics::View
- OnRemoveFromStage() : Graphics::Cell, Graphics::GraphicsEntity
- OnRemoveFromView() : Graphics::CameraEntity
- OnRender() : Graphics::GraphicsEntity, Game::Entity, Game::Property
- OnRenderDebug() : Graphics::GraphicsEntity, Graphics::ModelEntity, BaseGameFeature::EntityManager, Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property, PhysicsFeature::MouseGripperProperty
- OnRenderFrame() : App::RenderApplication, App::ViewerApplication
- OnReset() : Base::GamePadBase, Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- OnResetDevice() : Direct3D9::D3D9ShaderInstance
- OnRestored() : Win32::Win32DisplayDevice
- OnSave() : Base::StreamTextureSaverBase, Direct3D9::D3D9StreamTextureSaver, Resources::ResourceSaver, BaseGameFeature::EntityManager, Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property
- OnSetCursor() : Win32::Win32DisplayDevice
- OnSetFocus() : Win32::Win32DisplayDevice
- OnShow() : Graphics::GraphicsEntity
- OnStart() : BaseGameFeature::EntityManager, Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property
- OnStateEnter() : App::GameStateHandler
- OnStateLeave() : App::GameStateHandler
- OnToggleFullscreenWindowed() : Win32::Win32DisplayDevice
- OnUnregister() : Scripting::Command
- **OnUpdateFrame()**: `App::AsyncRenderApplication`, `App::AsyncViewerApplication`, `App::RenderApplication`

- **Open()**: `IO::ZipFileStream`, `IO::ZipArchive`, `Messaging::AsyncPort`, `Messaging::Handler`, `Net::TcpServer`, `Win32::Win32Socket`, `Scripting::LuaServer`, `Scripting::ScriptServer`, `App::AsyncRenderApplication`, `App::AsyncViewerApplication`, `App::RenderApplication`, `App::ViewerApplication`, `AsyncGraphics::AsyncGraphicsHandler`, `AsyncGraphics::AsyncGraphicsInterface`, `Resources::SharedResourceServer`, `AsyncGraphics::DisplayProxy`, `AsyncGraphics::GraphicsServerProxy`, `Base::DisplayDeviceBase`, `Models::ModelWriter`, `Base::RenderDeviceBase`, `Base::ShaderServerBase`, `Base::ShapeRendererBase`, `Base::TransformDeviceBase`, `Direct3D9::D3D9RenderDevice`, `Direct3D9::D3D9ShaderServer`, `Direct3D9::D3D9ShapeRenderer`, `Lighting::SM30LightServer`, `CoreGraphics::VertexLayoutServer`, `Win32::Win32DisplayDevice`, `Frame::FrameServer`, `Graphics::GraphicsServer`, `Base::InputServerBase`, `Win32::Win32InputServer`, `Lighting::LightServerBase`, `Lighting::SM30ShadowServer`, `Models::BinaryModelReader`, `Models::ModelReader`, `Models::N2ModelReader`, `Models::XmlModelReader`, `Models::ModelServer`, `Models::BinaryModelWriter`, `Models::XmlModelWriter`, `Models::VisResolver`, `Resources::ResourceManager`, `App::GameApplication`, `BaseGameFeature::LoaderServer`, `Game::GameServer`, `App::Application`, `App::ConsoleApplication`, `AsyncHttp::AsyncHttpInterface`, `Core::CoreServer`, `Http::HtmlPageWriter`, `Http::HttpServer`, `IO::BinaryReader`, `IO::BinaryWriter`, `IO::Console`, `IO::ConsoleHandler`, `IO::FileStream`, `IO::MemoryStream`, `IO::Stream`, `IO::StreamReader`, `IO::StreamWriter`, `IO::XmlReader`, `IO::XmlWriter`

- **OpenDInputMouse()**: `Win32::Win32InputServer`

- **OpenFile()**: `Win32::Win32FSWrapper`

- **OpenProgressIndicator()**: `BaseGameFeature::LoaderServer`
- OpenWindow() : Win32::Win32DisplayDevice
- operator *() : Math::vector, Math::float2, Util::List< TYPE >::Iterator
- operator *=( ) : Math::float2, Math::vector
- operator bool() : Util::List< TYPE >::Iterator
- operator delete() : Util::Blob, Util::String, Win32::Win32Guid
- operator new() : Util::Blob, Util::String, Win32::Win32Guid
- operator !=() : Attr::Attribute, Attr::AttrId, Attr::BlobAttrId, Attr::Float4AttrId, Attr::FloatAttrId, Attr::IntAttrId, Attr::Matrix44AttrId, Attr::StringAttrId, IO::MediaType, IO::URI, Win32::Win32FileTime, Math::point, Math::vector, Math::float2, Util::Array< TYPE >, Util::Blob< TYPE >, Util::FixedArray< TYPE >, Util::FourCC, Util::KeyValuePair< KEYTYPE, VALUETYPE >, Util::List< TYPE >::Iterator, Util::Variant, Util::Proxy< TYPE >, Util::Queue< TYPE >, Util::Stack< TYPE >, Util::String, Util::Variant, Win32::Win32Guid, CoreGraphics::DisplayMode, Util::FixedTable< TYPE >, Util::Variant, Util::Proxy< TYPE >, Attr::GuidAttrId, Util::Atom< TYPE >, Core::Rtti, Attr::BlobAttrId, Attr::Attribute
- operator+() : Math::vector, Math::float2, Math::point
- operator++() : Util::List< TYPE >::Iterator
- operator+=() : Math::vector, Math::point, Math::float2, Util::String
- operator-() : Math::vector, Math::float2, Math::point, Math::vector, Math::float2
- operator--() : Util::List< TYPE >::Iterator
- operator-() : Math::point, Math::vector, Math::float2
- operator->() : Util::List< TYPE >::Iterator
- operator<() : Util::KeyValuePair< KEYTYPE, VALUETYPE >, Util::Proxy< TYPE >, Util::Blob, Attr::AttrId, Util::Atom< TYPE >, Util::FourCC, Util::Proxy< TYPE >, Win32::Win32FileTime, Win32::Win32IpAddress, Util::Atom< TYPE >, Util::String, Win32::Win32Guid
- operator<=() : Util::Blob, Util::Proxy< TYPE >, Win32::Win32Guid, Util::KeyValuePair< KEYTYPE, VALUETYPE >, Util::Atom< TYPE >, Attr::AttrId, Util::Atom< TYPE >, Util::FourCC, Util::String
operator=() : Util::Variant, Attr::Attribute, Math::point, Util::Variant, Math::vector, Math::float2, Attr::Attribute, IO::MediaType, Attr::Attribute, Util::String, Util::Variant, Util::KeyValuePair<KEYTYPE, VALUETYPE>, Util::Blob, Util::Variant, Util::String, Util::Variant, Attr::Attribute, Util::Proxy<TYPE>, Util::Variant, Util::Proxy<TYPE>, Util::Variant, Util::Stack<TYPE>, Util::Variant, Attr::Attribute, Util::Variant, Util::List<TYPE>, Attr::Attribute, IO::URI, Threading::SafePriorityQueue<PRITYPE, TYPE>, Math::polar, Util::Atom<TYPE>, Util::Variant, Util::FixedSizeArray<TYPE>, Util::Queue<TYPE>, Util::Variant, Threading::SafeQueue<TYPE>, Util::Variant, Win32::Win32Guid, Util::List<TYPE>::Iterator, Util::Array<TYPE>, Util::FixedSizeTable<TYPE>, Util::HashTable<KEYTYPE, VALUETYPE>, Util::Dictionary<KEYTYPE, VALUETYPE>, Util::Atom<TYPE>

operator==() : Util::Variant, Attr::Attribute, Attr::FloatAttrId, Util::Variant, Math::vector, Util::Variant, Util::Proxy<TYPE>, Attr::Attribute, Util::FixedSizeArray<TYPE>, Attr::Matrix44AttrId, Util::FourCC, Win32::Win32Guid, Util::List<TYPE>::Iterator, IO::URI, Math::float2, Util::Proxy<TYPE>, Math::point, Util::String, Core::Rtti, IO::MediaType, Util::KeyValuePair<KEYTYPE, VALUETYPE>, Attr::Float4AttrId, Util::Variant, Attr::GuidAttrId, Util::FixedSizeTable<TYPE>, Win32::Win32IpAddress, Util::Atom<TYPE>, Util::Variant, Util::Blob, Win32::Win32FileTime, Attr::Attribute, CoreGraphics::DisplayMode, Attr::Attribute, Util::Stack<TYPE>, Attr::StringAttrId, Attr::AttrId, Attr::BlobAttrId, Util::String, Util::Array<TYPE>, Attr::Attribute, Attr::Variant, Attr::Attribute, Util::Variant, Attr::Attribute, Util::Variant, Attr::Attribute, Attr::IntAttrId, Util::String, Util::Atom<TYPE>, Util::Variant, Util::Queue<TYPE>, Attr::BoolAttrId, Util::Variant, Messaging::Id

operator> : Win32::Win32FileTime, Util::FourCC, Win32::Win32Guid, Attr::AttrId, Util::Proxy<TYPE>, Util::Atom<TYPE>, Win32::Win32IpAddress, Util::KeyValuePair<KEYTYPE, VALUETYPE>, Util::Proxy<TYPE>, Util::Blob, Util::Atom<TYPE>, Util::String

operator>= : Util::String, Util::Atom<TYPE>,
Util::KeyValuePair<KEYTYPE, VALUETYPE>, Util::Proxy<TYPE>, Util::Atom<TYPE>, Win32::Win32Guid, Util::Blob, Util::FourCC, Util::Proxy<TYPE>, Attr::AttrId

- operator[]( ): Util::Dictionary<KEYTYPE, VALUETYPE>, Util::Array<TYPE>, Util::HashTable<KEYTYPE, VALUETYPE>, Util::SimpleTree<VALUETYPE>::Node, Util::FixedArray<TYPE>, Util::Stack<TYPE>, Util::Queue<TYPE>, Util::String, Util::SimpleTree<VALUETYPE>::Node, Util::Dictionary<KEYTYPE, VALUETYPE>, Util::String

- origin(): Math::point
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- p -

- Parent() : Util::SimpleTree< VALUETYPE >::Node
- ParseBluePrints() : BaseGameFeature::FactoryManager
- ParseQuery() : IO::URI
- ParticleSystemNode() : Models::ParticleSystemNode
- ParticleSystemNodeInstance() : 
  Models::ParticleSystemNodeInstance
- Peek() : Threading::SafePriorityQueue< PRITYPE, TYPE >, Threading::SafeQueue< TYPE >, Win32::Win32Event, Util::Queue< TYPE >, Util::Stack< TYPE >, Messaging::AsyncPort
- PerformGarbageCollection() : Util::Atom< TYPE >
- PhysicsProperty() : PhysicsFeature::PhysicsProperty
- point() : Math::point
- pointat() : Math::line
- polar() : Math::polar
- Pop() : Util::Stack< TYPE >
- Port() : IO::URI
- Position : IO::Stream
- Prepare() : Resources::ResourceManager
- Present() : Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
- PrimitiveGroup() : CoreGraphics::PrimitiveGroup
- Print() : IO::Console, IO::ConsoleHandler, Win32::Win32ConsoleHandler, IO::Console
- PrintCommandHelp() : Scripting::ScriptServer
- PrintCommandList() : Scripting::ScriptServer
- PrintDebug() : Attr::AttributeTable
- Priority : Resources::ManagedResource, Win32::Win32Thread
- ProcessWindowMessages() : Win32::Win32DisplayDevice, Base::DisplayDeviceBase
- project_screen_rh() : Math::sphere
- Property() : Game::Property
- Protocol : Win32::Win32Socket
- Proxy() : Util::Proxy< TYPE >
- PSSMUtil() : Lighting::PSSMUtil
- Push() : Util::Stack< TYPE >
- PutEvent() : Base::InputServerBase, CoreGraphics::DisplayEventHandler, CoreGraphics::RenderEventHandler, CoreGraphics::ThreadSafeDisplayEventHandler, CoreGraphics::ThreadSafeRenderEventHandler
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- q -

- QuadtreeStageBuilder() : [Graphics::QuadtreeStageBuilder](#)
- 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
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **r** -

- Read() : `IO::FileStream`, `IO::MemoryStream`, `Win32::Win32FSWrapper`, `IO::ZipFileStream`, `IO::Stream`, `IO::ZipFileEntry`, `System::Win32Registry`
- ReadAll() : `IO::TextReader`
- ReadAllLines() : `IO::TextReader`
- ReadBlob() : `IO::BinaryReader`
- ReadBool() : `IO::BinaryReader`
- ReadChar() : `IO::BinaryReader`, `IO::TextReader`
- ReadDInputMouse() : `Win32::Win32InputServer`
- ReadDouble() : `IO::BinaryReader`
- ReadFloat() : `IO::BinaryReader`
- ReadFloat4() : `IO::BinaryReader`
- ReadGuid() : `IO::BinaryReader`
- ReadInt() : `IO::BinaryReader`
- ReadLine() : `IO::TextReader`
- ReadMatrix44() : `IO::BinaryReader`
- ReadMessage() : `Messaging::MessageReader`
- ReadRequest() : `Http::HttpRequestReader`
- ReadShort() : `IO::BinaryReader`
- ReadString() : `IO::BinaryReader`
- ReadUChar() : `IO::BinaryReader`
- ReadUInt() : `IO::BinaryReader`
- ReadUShort() : `IO::BinaryReader`
- Realloc() : `Win32::Win32Heap`
- Reallocate() : `Util::Array< TYPE >`
- rectangle() : `Math::rectangle< TYPE >`
- `Recv()`: `Net::TcpClient`, `Net::TcpClientConnection`, `Win32::Win32Socket`
- `RecvFrom()`: `Win32::Win32Socket`
- `RefCounted()`: `Core::RefCounted`
- `Register()`: `Attr::AttributeDefinitionBase`, `Core::Factory`
- `RegisterCommand()`: `Scripting::LuaServer`, `Scripting::ScriptServer`
- `RegisterDynamicAttribute()`: `Attr::AttributeDefinitionBase`
- `RegisterMessage()`: `Messaging::Port`
- `RegisterPropertyCallback()`: `Game::Entity`
- `RegisterSharedResource()`: `Resources::SharedResourceServer`
- `RegisterUriScheme()`: `IO::IoServer`
- `Release()`: `Core::RefCounted`, `Util::SimpleTree< VALUETYPE, Core::RefCounted, Util::SimpleTree< VALUETYPE > >`
- `ReleaseKeyboardCapture()`: `Base::InputServerBase`
- `ReleaseMouseCapture()`: `Base::InputServerBase`
- `Remove()`: `Util::List< TYPE >`
- `RemoveAllLoaders()`: `BaseGameFeature::LoaderServer`
- `RemoveAllMappers()`: `Resources::ResourceManager`
- `RemoveBack()`: `Util::List< TYPE >`
- `RemoveDisplayEventHandler()`: `AsyncGraphics::DisplayProxy`
- `RemoveEntity()`: `Graphics::Cell`, `Graphics::Stage`, `BaseGameFeature::EntityManager`
- `RemoveEntityFromTriggered()`: `BaseGameFeature::EntityManager`
- `RemoveEntityImmediate()`: `BaseGameFeature::EntityManager`
- `RemoveEntityLoader()`: `BaseGameFeature::LoaderServer`
- `RemoveEventHandler()`: `Base::DisplayDeviceBase`, `Base::RenderDeviceBase`
- `RemoveFront()`: `Util::List< TYPE >`
- `RemoveGameFeature()`: `Game::GameServer`
- `RemoveHandler()`: `IO::Console`, `Messaging::Port`
- `RemoveInputHandler()`: `Base::InputServerBase`
- `RemoveManager()`: `Game::FeatureUnit`
- `RemoveMapper()`: `Resources::ResourceManager`
- `RemoveNode()`: `Models::Model`
- `RemoveNullEntriesFromArrays()`: `BaseGameFeature::EntityManager`
- RemovePort() : Messaging::Dispatcher
- RemovePreShader() : Base::ShaderInstanceBase
- RemoveRenderEventHandler() : AsyncGraphics::DisplayProxy
- RemoveRequestHandler() : Http::HttpServer
- RenameLevel() : BaseGameFeature::CategoryManager
- Render() : Frame::FrameBatch, Frame::FramePostEffect, Frame::FrameShader, Graphics::View, Models::ModelNodeInstance, Models::ParticleSystemNodeInstance, Models::ShapeNodeInstance, Models::SkinShapeNodeInstance, Frame::FramePass
- RenderApplication() : App::RenderApplication
- RenderDebug() : Graphics::View, Models::ModelInstance, Models::ModelNodeInstance, Game::GameServer, Models::CharacterNodeInstance
- RenderDebugSimple() : Graphics::View
- RenderDevice() : CoreGraphics::RenderDevice
- RenderDeviceBase() : Base::RenderDeviceBase
- RenderEvent() : CoreGraphics::RenderEvent
- RenderEventHandler() : CoreGraphics::RenderEventHandler
- RenderFragment() : Models::SkinShapeNodeInstance
- RenderSkinning() : Models::SkinShapeNodeInstance
- RenderTargetBase() : Base::RenderTargetBase
- ReplaceChars() : Util::String
- ReplaceIllegalFilenameChars() : Util::String
- requestedState : App::GameApplication
- RequestLoadResources() : Models::SkinShapeNode
- RequestQuit() : Game::GameServer
- RequestState() : App::GameApplication
- RequestUnloadResources() : Models::SkinShapeNode
- Reserve() : Util::Array< TYPE >, Util::String, Util::Blob
- ReserveRows() : Attr::AttributeTable
- Reset() : RenderUtil::MayaCameraUtil, Base::InputServerBase, Models::VisResolveContainer< TYPE >, Util::Array< TYPE >, Win32::Win32Timer
- ResetFeatureBits() : Base::ShaderServerBase
- ResetModifiedState() : Attr::AttributeTable
- ResolveAssigns() : IO::IoServer
- ResolveAssignsInString() : IO::IoServer
- ResolveVisibleLights() : Graphics::View
- ResolveVisibleModelNodeInstances() : Graphics::View
- Resource() : Resources::Resource
- ResourceBase() : Base::ResourceBase
- ResourceLoader() : Resources::ResourceLoader
- ResourceManager() : Resources::ResourceManager
- ResourceMapper() : Resources::ResourceMapper
- ResourceSaver() : Resources::ResourceSaver
- Root() : Util::SimpleTree< VALUETYPE >
- RootKey : System::Win32Registry
- Row() : Util::QuadTree< TYPE >::Node
- RowIndex() : BaseGameFeature::CategoryManager::Entry
- Rtti() : Core::Rtti
- Run() : App::Application, App::GameApplication, App::AsyncRenderApplication, App::RenderApplication
- Running() : Win32::Win32Timer

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **S** -

- SafePriorityQueue() : Threading::SafePriorityQueue<PRITYPE, TYPE>
- SafeQueue() : Threading::SafeQueue<TYPE>
- sampledCurves : Models::ParticleSystemNode
- Save() : Resources::Resource, BaseGameFeature::UserProfile
- SaveAttributes() : BaseGameFeature::GlobalAttrsManager
- SaveModel() : Models::ModelServer
- SaveScreenshot() : Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
- SaveToAttrs() : Models::Model, Models::ModelNode, Models::TransformNode
- Scheme() : IO::URI
- ScriptingPageHandler() : Debug::ScriptingPageHandler
- ScriptServer() : Scripting::ScriptServer
- Seek() : IO::FileStream, IO::MemoryStream, IO::Stream, Win32::Win32FSWrapper, IO::ZipFileStream
- SeekOrigin : IO::Stream
- SelectActiveVariation() : Base::ShaderInstanceBase, Direct3D9::D3D9ShaderInstance
- Semantic : Base::ShaderVariableBase
- SemanticName : CoreGraphics::VertexComponent
- SemanticNameToString() : CoreGraphics::VertexComponent
- Send() : Net::TcpClientConnection, Win32::Win32Socket, Messaging::AsyncPort, Messaging::Port, Net::TcpClient
- SendCreateMsg() : AsyncGraphics::GraphicsEntityProxy
- `SendMsg()` : `AsyncGraphics::GraphicsEntityProxy`
- `SendSync()` : `Game::Entity`
- `SendTo()` : `Win32::Win32Socket`
- `SendWait()` : `Messaging::AsyncPort`
- `Set()` : `IO::MediaType , IO::URI , Util::Array< TYPE > , Util::Blob , Util::FixedTable< TYPE > , Util::String`
- `set()` : `Math::bbox , Math::point , Math::vector , Math::float2 , Math::line , Math::polar , Math::rectangle< TYPE > , Math::sphere`
- `SetAbsMousePos()` : `Input::InputEvent`
- `SetAccess()` : `Base::ResourceBase`
- `SetAccessMode()` : `IO::Stream`
- `SetAccessPattern()` : `IO::Stream`
- `SetActiveShaderInstance()` : `Base::ShaderServerBase`
- `SetActivityDistance()` : `Models::ParticleSystemNode`
- `SetAdapter()` : `AsyncGraphics::DisplayProxy , Base::DisplayDeviceBase`
- `SetAddress()` : `Net::TcpServer , Win32::Win32Socket`
- `SetAllNodeInstancesVisible()` : `Models::ModelInstance`
- `SetAlwaysOnTop()` : `AsyncGraphics::DisplayProxy , Base::DisplayDeviceBase`
- `SetAnim()` : `Models::CharacterNode`
- `SetAnimationMapping()` : `Graphics::ActorEntity`
- `SetAntiAliasQuality()` : `AsyncGraphics::DisplayProxy , Base::DisplayDeviceBase , Base::RenderTargetBase`
- `SetAppName()` : `App::Application , Core::CoreServer`
- `SetAssign()` : `IO::IoServer`
- `SetAsyncEnabled()` : `Resources::Resource , Resources::ResourceMapper`
- `SetAttr()` : `Attr::AttributeContainer , Attr::AttributeTable , Models::Model , Models::ModelNode , Game::Entity`
- `SetAttributes()` : `Graphics::StageBuilder`
- `SetAttrId()` : `Attr::Attribute`
- `SetBackLightColor()` : `Lighting::GlobalLightEntity`
- `SetBaseAnimation()` : `Graphics::ActorEntity`
- `SetBaseIndex()` : `CoreGraphics::PrimitiveGroup`
- `SetBaseVertex()` : `CoreGraphics::PrimitiveGroup`
- `SetBillboardOrientation()` : `Models::ParticleSystemNode`
- `SetBlob()` : `Attr::Attribute , Attr::AttributeContainer`
Attr::AttributeTable, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::GlobalAttrsManager, Game::Entity
- SetBlobArray() : Util::Variant
- SetBlocking() : Net::TcpClient, Win32::Win32Socket
- SetBool() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlWriter, Util::String, Util::Variant, Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- SetBoolArray() : Util::Variant, Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable
- SetBoundingBox() : CoreGraphics::PrimitiveGroup, Graphics::Cell, Models::Model, Models::ModelNode
- SetBroadcast() : Win32::Win32Socket
- SetCameraEntity() : Graphics::View, Lighting::PSSMUtil
- SetCameraFocusEntity() : BaseGameFeature::FocusManager
- SetCameraFocusToNextEntity() : BaseGameFeature::FocusManager
- SetCastShadows() : Lighting::AbstractLightEntity
- SetChar() : Input::InputEvent
- SetCharPtr() : Util::String
- SetChildrenVisibility() : Models::ModelNodeInstance
- SetClearColor() : Base::RenderTargetBase, Frame::FramePass
- SetClearDepth() : Base::RenderTargetBase, Frame::FramePass
- SetClearStencil() : Base::RenderTargetBase, Frame::FramePass
- SetClip() : Models::CharacterNode
- SetCmdLineArgs() : App::Application, Game::FeatureUnit
- SetColor() : Lighting::AbstractLightEntity
- SetCompanyName() : App::Application, Core::CoreServer
- SetContent() : Http::HttpResponseWriter
- SetCoreId() : Win32::Win32Thread
- SetCurve() : Models::ParticleSystemNode
- SetDataPtr() : Util::QuadTree< TYPE >::Node
- SetDay() : Base::CalendarTimeBase
- SetDebugTextEnabled() : BaseGameFeature::LoaderServer
- SetDefaultRenderTarget() : Base::RenderTargetBase
- SetDefaultView() : Graphics::GraphicsServer
- SetDepth() : Base::TextureBase
- SetDesc() : Http::HttpRequestHandler
- SetDescription() : CoreGraphics::AdapterInfo
- SetDeviceId() : CoreGraphics::AdapterInfo
- SetDeviceName() : CoreGraphics::AdapterInfo
- SetDisplayMode() : AsyncGraphics::DisplayProxy , Base::DisplayDeviceBase
- SetDisplayModeSwitchEnabled() :
  AsyncGraphics::DisplayProxy , Base::DisplayDeviceBase
- SetDistance() : BaseGameFeature::MoveFollow
- SetDontLinger() : Win32::Win32Socket
- SetDriverName() : CoreGraphics::AdapterInfo
- SetDriverVersionHighPart() : CoreGraphics::AdapterInfo
- SetDriverVersionLowPart() : CoreGraphics::AdapterInfo
- SetEmissionDuration() : Models::ParticleSystemNode
- SetEmitter() : Models::ParticleSystemNodeInstance
- SetEnabled() : PhysicsFeature::PhysicsProperty
- SetEntities() : GraphicsFeature::GetGraphicsEntities
- SetEntity() : Game::Property ,
  GraphicsFeature::GetGraphicsEntities
- SetEnvEntityTransform() :
  BaseGameFeature::EnvEntityManager
- SetError() : Scripting::Command , Scripting::ScriptServer
- SetFadeAnimationMix() : Graphics::ActorEntity
- SetFeatureBits() : Base::ShaderServerBase
- SetFeatureMask() : Base::ShaderVariationBase
- SetFloat() : Attr::Attribute , Attr::AttributeContainer ,
  Attr::AttributeTable , IO::XmlWriter , Util::String , Util::Variant ,
  Base::ShaderVariableBase ,
  Base::ShaderVariableInstanceBase ,
  Direct3D9::D3D9ShaderVariable , Models::Model ,
  Models::ModelNode , BaseGameFeature::UserProfile ,
  BaseGameFeature::GlobalAttrsManager , Game::Entity
- SetFloat4() : Attr::Attribute , Attr::AttributeContainer ,
Attr::AttributeTable, IO::XmlWriter, Util::String, Util::Variant,
Models::Model, Models::ModelNode,
BaseGameFeature::UserProfile,
BaseGameFeature::GlobalAttrsManager, Game::Entity

- SetFloat4Array() : Util::Variant
- SetFloatArray() : Util::Variant, Base::ShaderVariableBase,
  Base::ShaderVariableInstanceBase,
  Direct3D9::D3D9ShaderVariable
- SetFocusEntity() : BaseGameFeature::FocusManager
- SetFocusToNextEntity() : BaseGameFeature::FocusManager
- SetFormat() : Base::StreamTextureSaverBase
- SetFragGroupIndex() : Models::SkinShapeNode
- SetFragment() : IO::URI
- SetFrameShader() : Graphics::View
- SetFromString() : Util::FourCC
- SetFromUInt() : Util::FourCC
- SetFromURI() : Interface::CopyFile
- SetFullscreen() : AsyncGraphics::DisplayProxy,
  Base::DisplayDeviceBase
- SetGravity() : Models::ParticleSystemNode
- SetGuid() : Attr::Attribute, Attr::AttributeContainer,
  Attr::AttributeTable, Util::Variant,
  CoreGraphics::AdapterInfo, Models::Model,
  Models::ModelNode,
  BaseGameFeature::GlobalAttrsManager, Game::Entity
- SetGuidArray() : Util::Variant
- SetHandled() : Messaging::Message
- SetHeight() : Base::RenderTargetBase, Base::TextureBase,
  CoreGraphics::DisplayMode
- SetHighFrequencyVibrator() : Base::GamePadBase
- SetHorizontalRotation() : GraphicsFeature::CameraOrbit
- SetHost() : IO::URI
- SetHostName() : Win32::Win32IpAddress
- SetHour() : Base::CalendarTimeBase
- SetIconName() : AsyncGraphics::DisplayProxy,
  Base::DisplayDeviceBase
- SetIndexAccessMode() :
  Resources::DynamicMeshResourceLoader
- SetIndexBuffer() : Base::MeshBase, Base::RenderDeviceBase
Direct3D9::D3D9RenderDevice
- SetIndexBufferDepth(): CoreGraphics::CPUIndexBuffer
- SetIndexBufferType(): Resources::DynamicMeshResourceLoader
- SetIndexData(): Resources::DynamicMeshResourceLoader
- SetIndexType(): Base::IndexBufferBase
- SetIndexUsage(): Resources::DynamicMeshResourceLoader
- SetInputFocusEntity(): BaseGameFeature::FocusManager
- SetInputFocusToNextEntity(): BaseGameFeature::FocusManager
- SetInstanceEntity(): BaseGameFeature::CategoryManager
- SetInt(): Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlWriter, Util::String, Util::Variant, Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- SetIntArray(): Util::Variant, Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable
- SetInvisible(): Models::ParticleSystemNode
- SetJoint(): Models::CharacterNode
- SetJointIndex(): Models::SkinShapeNode
- SetJointIndices(): Models::SkinShapeNode
- SetKeepAlive(): Win32::Win32Socket
- SetKey(): Input::InputEvent
- SetLevelName(): App::GameStateHandler
- SetLightDir(): Lighting::PSSMUtil
- SetLightingMode(): Frame::FrameBatch
- SetLightType(): Lighting::AbstractLightEntity
- SetLoader(): Resources::Resource
- SetLocalBoundingBox(): Graphics::GraphicsEntity
- SetLocalPath(): IO::URI
- SetLoop(): Models::ParticleSystemNode
- SetLowFrequencyVibrator(): Base::GamePadBase
- SetMainRenderTarget(): Frame::FrameShader
- SetManagedResourceClass(): Resources::SimpleResourceMapper
- SetMatrix() : Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable
- SetMatrix44() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlWriter, Util::String, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::GlobalAttrsManager, Game::Entity
- SetMatrix44Array() : Util::Variant
- SetMatrixArray() : Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable
- SetMaxProgressValue() : BaseGameFeature::LoaderServer
- SetMaxTriggerDistance() : BaseGameFeature::EntityManager
- SetMediaType() : IO::Stream
- SetMemoryMappingEnabled() : IO::BinaryReader, IO::BinaryWriter
- SetMethod() : Http::HttpRequest
- SetMilliSecond() : Base::CalendarTimeBase
- SetMinute() : Base::CalendarTimeBase
- SetMipLevel() : Base::StreamTextureSaverBase
- SetMipMapsEnabled() : Base::RenderTargetBase
- SetModel() : Models::ModelReader
- SetModelEntity() : Models::ModelInstance
- SetModelResId() : Models::ModelReader
- SetModelResourceMapper() : Models::ModelServer
- SetModelTransform() : Base::TransformDeviceBase
- SetModifiedTracking() : Attr::AttributeTable
- SetMonth() : Base::CalendarTimeBase
- SetMouseButton() : Input::InputEvent
- SetMouseExcludeSet() : BaseGameFeature::EnvQueryManager
- SetMouseMovement() : RenderUtil::MayaCameraUtil
- SetName() : Http::HttpRequestHandler, Messaging::AsyncPort, Win32::Win32Thread, Base::ShaderVariableBase, Base::ShaderVariationBase, Frame::FramePass, Frame::FramePostEffect, Frame::FrameShader, Graphics::Stage, Graphics::View, Models::ModelNode, BaseGameFeature::UserProfile
- SetNodeFilter() : Frame::FrameBatch
- SetNoDelay() : `Win32::Win32Socket`
- SetNormMousePos() : `Input::InputEvent`
- SetNumArrayElements() : `Base::ShaderVariableBase`
- SetNumberOfIndices() : `Resources::DynamicMeshResourceLoader`
- SetNumberOfVertices() : `Resources::DynamicMeshResourceLoader`
- SetNumIndices() : `Base::IndexBufferBase`, `CoreGraphics::PrimitiveGroup`
- SetNumberOfMipLevels() : `Base::TextureBase`
- SetNumPasses() : `Base::ShaderVariationBase`
- SetNumVertices() : `Base::VertexBufferBase`, `CoreGraphics::PrimitiveGroup`
- SetObject() : `Util::Variant`
- SetObtainFocus() : `GraphicsFeature::CameraFocus`, `GraphicsFeature::InputFocus`
- SetOrbitButton() : `RenderUtil::MayaCameraUtil`
- SetOrbiting() : `RenderUtil::MayaCameraUtil`
- SetOverlayAnimation() : `Graphics::ActorEntity`
- SetPanButton() : `RenderUtil::MayaCameraUtil`
- SetPanning() : `RenderUtil::MayaCameraUtil`
- SetParent() : `Models::ModelNode`, `Models::ModelNodeInstance`
- SetParticleRotationRandomize() : `Models::ParticleSystemNode`
- SetParticleSizeRandomize() : `Models::ParticleSystemNode`
- SetParticleStretch() : `Models::ParticleSystemNode`
- SetParticleVelocityRandomize() : `Models::ParticleSystemNode`
- SetPassword() : `IO::ZipArchive`
- SetPixelFormat() : `CoreGraphics::DisplayMode`, `Base::TextureBase`
- SetPlaceholder() : `Resources::ManagedResource`
- SetPlaceholderResourceId() : `Resources::ResourceMapper`
- SetPort() : `IO::URI`, `Http::HttpServer`, `Win32::Win32IpAddress`
- SetPosition() : `Models::TransformNodeInstance`
- setPosition() : `Math::transform44`
- SetPosition() : `Models::TransformNode`
- SetPrecalcTime() : `Models::ParticleSystemNode`
- SetPrimitiveGroup() : `Base::RenderDeviceBase`
- SetPrimitiveGroups() : `Base::MeshBase`
- SetPrimitiveTopology() : CoreGraphics::PrimitiveGroup
- SetPriority() : Win32::Win32Thread, Resources::ManagedResource
- SetProgressResource() : BaseGameFeature::LoaderServer
- SetProgressText() : BaseGameFeature::LoaderServer
- SetProjMapUvOffsetAndScale() : Lighting::AbstractLightEntity
- SetProjTransform() : Base::TransformDeviceBase
- SetQuadTreeSettings() : Graphics::QuadtreeStageBuilder
- SetQuery() : IO::URI
- SetQuitRequested() : App::AsyncRenderApplication, App::RenderApplication, Base::InputServerBase
- SetRandomRotDir() : Models::ParticleSystemNode
- SetReadOnly() : IO::IoServer, Win32::Win32FSWrapper
- SetRecvBufSize() : Win32::Win32Socket
- SetRelativeDistanceChange() : GraphicsFeature::CameraDistance
- SetRenderOldestFirst() : Models::ParticleSystemNode
- SetRenderTarget() : Graphics::View, Frame::FramePass, Frame::FramePostEffect
- SetResolved() : Models::VisResolveContainer<TYPE>
- SetResolveRect() : Base::RenderTargetBase
- SetResolveTextureHeight() : Base::RenderTargetBase
- SetResolveTextureResourceId() : Base::RenderTargetBase
- SetResolveTextureWidth() : Base::RenderTargetBase
- SetResource() : Resources::ManagedResource
- SetResourceClass() : Resources::SimpleResourceMapper
- SetResourceId() : Graphics::ModelEntity, Resources::ManagedResource, Resources::Resource
- SetResourceLoaderClass() : Resources::SimpleResourceMapper
- SetResourceType() : Resources::ManagedResource
- SetResponseContentStream() : Http::HttpRequest
- SetResult() : Interface::CopyFile, Interface::IOMessage
- SetReturnCode() : App::Application
- SetReuseAddr() : Win32::Win32Socket
- SetRevision() : CoreGraphics::AdapterInfo
- SetRgbCurve() : Models::ParticleSystemNode
- SetRootCell() : Graphics::Stage
- SetRootLocation() : Http::HttpRequestHandler
- SetRotate() : Models::TransformNodeInstance
- setrotate() : Math::transform44
- SetRotate() : Models::TransformNode
- setrotatepivot() : Math::transform44
- SetRotatePivot() : Models::TransformNode, Models::TransformNodeInstance
- SetRowUserData() : Attr::AttributeTable
- SetSaveGame() : App::GameStateHandler
- SetSaver() : Resources::Resource
- SetScale() : Models::TransformNode
- setscale() : Math::transform44
- SetScale() : Models::TransformNodeInstance
- SetScalePivot() : Models::TransformNodeInstance
- setscalepivot() : Math::transform44
- SetScalePivot() : Models::TransformNode
- SetScheme() : IO::URI
- SetSecond() : Base::CalendarTimeBase
- SetSemantic() : Base::ShaderVariableBase
- SetSendBufSize() : Win32::Win32Socket
- SetServerAddress() : Net::TcpClient
- SetSetupMode() : App::GameStateHandler
- SetShader() : Frame::FrameBatch, Frame::FramePass, Frame::FramePostEffect
- SetShaderFeatures() : Frame::FrameBatch
- SetShaderParamBindMode() : Base::ShaderServerBase
- SetShadowBufferUvOffsetAndScale() : Lighting::AbstractLightEntity
- SetSize() : Util::FixedTable< TYPE >, IO::MemoryStream, IO::Stream, Util::FixedArray< TYPE >
- SetSkinList() : Graphics::ActorEntity
- SetSockAddr() : Win32::Win32IpAddress
- SetSortingMode() : Frame::FrameBatch
- SetStackSize() : Win32::Win32Thread
- SetStage() : Graphics::View
- SetStageBuilder() : Graphics::Stage
- SetStartDelay() : Models::ParticleSystemNode
- SetStartRotationMax() : Models::ParticleSystemNode
- SetStartRotationMin() : Models::ParticleSystemNode
- SetState() : Resources::Resource,
Resources::ResourceLoader, App::GameApplication
- SetStatus() : Http::HttpRequest
- SetStatusCode() : Http::HttpResponseWriter
- SetStream() : Interface::ReadStream, IO::StreamReader, IO::StreamWriter, Messaging::MessageReader, Base::StreamTextureSaverBase, Messaging::MessageWriter, Interface::WriteStream
- SetStreamByteOrder() : IO::BinaryReader, IO::BinaryWriter
- SetStretchDetail() : Models::ParticleSystemNode
- SetStretchToStart() : Models::ParticleSystemNode
- SetString() : Models::Model, Game::Entity, BaseGameFeature::GlobalAttrsManager, IO::XmlWriter, BaseGameFeature::UserProfile, Attr::AttributeContainer, Attr::AttributeTable, Attr::Attribute, Util::Variant, Models::ModelNode, Attr::AttributeTable
- SetStringArray() : Util::Variant
- SetSubSystemId() : CoreGraphics::AdapterInfo
- SetTargetEntityId() : BaseGameFeature::MoveFollow
- SetTexture() : Base::ShaderVariableBase, Direct3D9::D3D9ShaderVariable, Base::ShaderVariableInstanceBase
- SetThreadPriority() : Messaging::AsyncPort
- SetThreadStackSize() : Messaging::AsyncPort
- SetTileTexture() : Models::ParticleSystemNode
- SetTime() : Graphics::GraphicsEntity, Models::ModelInstance
- setTitle() : Http::HtmlPageWriter
- SetToDefault() : BaseGameFeature::UserProfile
- SetToFirstChild() : IO::XmlReader
- SetToNextChild() : IO::XmlReader
- SetToNode() : IO::XmlReader
- SetToParent() : IO::XmlReader
- SetToURI() : Interface::CopyFile
- SetTransform() : AsyncGraphics::GraphicsEntityProxy, Models::ModelInstance, Graphics::GraphicsEntity
- SetTripleBufferingEnabled() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- SetType() : Models::ModelNode, AsyncGraphics::GraphicsEntityProxy, Graphics::GraphicsEntity, Base::TextureBase, Util::Variant,
Frame::FrameBatch, Base::ShaderVariableBase, Input::InputEvent

- Setup(): Win32::Win32Heap, Util::Blob, Util::String, AsyncGraphics::CameraEntityProxy, Base::MemoryIndexBufferLoaderBase, Base::MemoryVertexBufferLoaderBase, Base::RenderTargetBase, Direct3D9::D3D9ShaderInstance, Direct3D9::D3D9VertexLayout, Frame::FramePostEffect, Win32::SysFunc, Win32::Win32Guid, RenderUtil::MayaCameraUtil, Base::MemoryIndexBufferLoaderBase, Base::VertexLayoutBase, Base::MemoryVertexBufferLoaderBase, Direct3D9::D3D9RenderTarget, Base::ShaderInstanceBase, Base::MemoryIndexBufferLoaderBase, AsyncGraphics::GraphicsEntityProxy

- SetupAcceptedMessages(): Messaging::Port, PhysicsFeature::PhysicsProperty

- SetupCallbacks(): PhysicsFeature::PhysicsProperty, PhysicsFeature::MouseGripperProperty, Game::Property

- SetupDefaultAttributes():
  PhysicsFeature::MouseGripperProperty, Game::Property, PhysicsFeature::PhysicsProperty

- SetupFromD3D9CubeTexture(): Direct3D9::D3D9Texture

- SetupFromD3D9Texture(): Direct3D9::D3D9Texture

- SetupFromD3D9VolumeTexture(): Direct3D9::D3D9Texture

- SetupGameFeatures(): App::GameApplication

- SetupManagedTextureVariable(): Models::StateNode

- SetupMode: App::GameStateHandler

- SetupMultiSampleType(): Direct3D9::D3D9RenderTarget

- SetupOrthogonal(): Graphics::CameraEntity, AsyncGraphics::CameraEntityProxy

- SetupPerspectiveFov(): Graphics::CameraEntity, AsyncGraphics::CameraEntityProxy

- SetupSkinInfos(): Models::CharacterNode

- SetupStateHandlers(): App::GameApplication

- SetURI(): Interface::IOMessage, IO::ZipArchive,
Http::HttpRequest, IO::Stream

- SetUsage(): Base::ResourceBase
- SetUserInfo(): IO::URI
- SetUserProfile(): BaseGameFeature::LoaderServer
- SetValid(): Graphics::GraphicsEntity
- SetValue(): Attr::Attribute
- SetValueFromString(): Attr::Attribute
- SetVariation(): Models::CharacterNode
- SetVariationsUri(): Models::CharacterNode
- SetVector(): Base::ShaderVariableBase, Direct3D9::D3D9ShaderVariable, Base::ShaderVariableInstanceBase
- SetVectorArray(): Base::ShaderVariableBase, Direct3D9::D3D9ShaderVariable, Base::ShaderVariableInstanceBase
- SetVendorId(): CoreGraphics::AdapterInfo
- SetVertexAccessMode(): Resources::DynamicMeshResourceLoader
- SetVertexBuffer(): Base::MeshBase, Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
- SetVertexComponents(): Resources::DynamicMeshResourceLoader
- SetVertexData(): Resources::DynamicMeshResourceLoader
- SetVertexLayout(): Base::VertexBufferBase
- SetVertexUsage(): Resources::DynamicMeshResourceLoader
- SetVertexWidth(): Resources::DynamicMeshResourceLoader
- SetVerticalRotation(): GraphicsFeature::CameraOrbit
- SetVerticalSyncEnabled(): Base::DisplayDeviceBase, AsyncGraphics::DisplayProxy
- SetViewAngleFade(): Models::ParticleSystemNode
- SetViewName(): AsyncGraphics::CameraEntityProxy
- SetViewTransform(): Base::TransformDeviceBase
- SetVisible(): Models::SkinShapeNodeInstance, Models::ModelNodeInstance, Graphics::GraphicsEntity, AsyncGraphics::GraphicsEntityProxy
- SetWaitForMessages(): Messaging::AsyncPort
- SetWeekday(): Base::CalendarTimeBase
- SetWidth(): Base::TextureBase, Base::RenderTargetBase, CoreGraphics::DisplayMode
- SetWindowTitle() : Base::DisplayDeviceBase
- SetXPos() : CoreGraphics::DisplayMode
- SetYear() : Base::CalendarTimeBase
- SetYPos() : CoreGraphics::DisplayMode
- SetZipFileSystemEnabled() : IO::IoServer
- SetZoomButton() : RenderUtil::MayaCameraUtil
- SetZoomIn() : RenderUtil::MayaCameraUtil
- SetZoomInButton() : RenderUtil::MayaCameraUtil
- SetZoomOut() : RenderUtil::MayaCameraUtil
- SetZoomOutButton() : RenderUtil::MayaCameraUtil
- ShaderBase() : Base::ShaderBase
- ShaderFeature() : CoreGraphics::ShaderFeature
- ShaderInstanceBase() : Base::ShaderInstanceBase
- ShaderPageHandler() : Debug::ShaderPageHandler
- ShaderParamBindMode : Base::ShaderServerBase
- 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
- ShapeType : Base::ShapeRendererBase
- SharedResourceServer() : Resources::SharedResourceServer
- Shutdown() : Net::TcpClientConnection
- Signal() : Threading::SafePriorityQueue< PRITYPE, TYPE >, Win32::Win32Event, Threading::SafeQueue< TYPE >
- SimpleResourceMapper() :
  Resources::SimpleResourceMapper
- SimpleStageBuilder() : Graphics::SimpleStageBuilder
- SimpleTree() : Util::SimpleTree< VALUETYPE >
- Size() : Util::HashTable< KEYTYPE, VALUETYPE >, Util::Blob
  , Util::Array< TYPE >, Util::List< TYPE >, Threading::SafeQueue< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::Dictionary< KEYTYPE,
VALUETYPE >, Util::FixedArray< TYPE >, Util::Stack< TYPE >
- size() : Math::bbox
- Size() : Util::Queue< TYPE >, Threading::SafePriorityQueue< PRITYPE, TYPE >
- SizeOf() : CoreGraphics::IndexType
- SkinShapeNode() : Models::SkinShapeNode
- SkinShapeNodeInstance() : Models::SkinShapeNodeInstance
- Sleep() : Win32::SysFunc
- SM30LightServer() : Lighting::SM30LightServer
- SM30ShadowServer() : Lighting::SM30ShadowServer
- Sort() : Util::Array< TYPE >, Util::FixedArray< TYPE >
- SortIfDirty() : Util::Dictionary< KEYTYPE, VALUETYPE >
- sphere() : Math::sphere
- SpotLightEntity() : Lighting::SpotLightEntity
- Stack() : Util::Stack< TYPE >
- Stage() : Graphics::Stage
- StageBuilder() : Graphics::StageBuilder
- start() : Math::line
- Start() : Game::GameServer, Win32::Win32Timer, Win32::Win32Thread
- StartEntities() : BaseGameFeature::EntityManager
- State : Resources::Resource
- StateNode() : Models::StateNode
- StateNodeInstance() : Models::StateNodeInstance
- Stop() : Win32::Win32Thread, Win32::Win32Timer, Game::GameServer
- StopOverlayAnimation() : Graphics::ActorEntity
- Stream() : IO::Stream
- StreamAnimationLoader() : CoreGraphics::StreamAnimationLoader
- StreamMeshLoader() : CoreGraphics::StreamMeshLoader
- StreamModelLoader() : Models::StreamModelLoader
- StreamReader() : IO::StreamReader
- StreamTextureSaverBase() : Base::StreamTextureSaverBase
- StreamWriter() : IO::StreamWriter
- String() : Util::String
- StringAttrId() : Attr::StringAttrId
- StringToMonth() : Base::CalendarTimeBase
- `StringToType()` : `Util::Variant`
- `StringToValueType()` : `Attr::Attribute`
- `StringToWeekday()` : `Base::CalendarTimeBase`
- `Strip()` : `Util::String`
- `StripFileExtension()` : `Util::String`
- `SubstituteChar()` : `Util::String`
- `SubstituteString()` : `Util::String`
- `SupportsDisplayMode()` : `Direct3D9::D3D9DisplayDevice`, `Base::DisplayDeviceBase`, `AsyncGraphics::DisplayProxy`
- `SwitchFocusEntities()` : `BaseGameFeature::FocusManager`
- `SystemTimeToFileTime()` : `Win32::Win32CalendarTime`, `Base::CalendarTimeBase`
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- \textbf{t} -

- \texttt{TcpClient()}: \texttt{Net::TcpClient}
- \texttt{TcpClientConnection()}: \texttt{Net::TcpClientConnection}
- \texttt{TcpServer()}: \texttt{Net::TcpServer}
- \texttt{Tell()}: \texttt{Win32::Win32FSWrapper}
- \texttt{TerminateAtIndex()}: \texttt{Util::String}
- \texttt{Text()}: \texttt{Http::HtmlPageWriter}
- \texttt{TextureBase()}: \texttt{Base::TextureBase}
- \texttt{TexturePageHandler()}: \texttt{Debug::TexturePageHandler}
- \texttt{ThreadSafeDisplayEventHandler()}: \texttt{CoreGraphics::ThreadSafeDisplayEventHandler}
- \texttt{ThreadSafeRenderEventHandler()}: \texttt{CoreGraphics::ThreadSafeRenderEventHandler}
- \texttt{ThreadStopRequested()}: \texttt{Win32::Win32Thread}
- \texttt{to\_matrix44()}: \texttt{Math::bbox}
- \texttt{ToHtml()}: \texttt{Http::HtmlElement}
- \texttt{ToHumanReadableString()}: \texttt{Http::HttpStatus}
- \texttt{Tokenize()}: \texttt{Util::String}
- \texttt{ToLower()}: \texttt{Util::String}
- \texttt{ToMediaType()}: \texttt{CoreGraphics::ImageFileFormat}
- \texttt{ToString()}: \texttt{Input::MouseButton}, \texttt{Models::ModelNodeType}, \texttt{Http::HttpMethod}, \texttt{Input::Key}, \texttt{Http::HttpStatus}, \texttt{Util::FourCC}, \texttt{CoreGraphics::Adapter}, \texttt{CoreGraphics::PixelFormat}, \texttt{CoreGraphics::AntiAliasQuality}, \texttt{CoreGraphics::BatchType}, \texttt{CoreGraphics::ImageFileFormat}, \texttt{CoreGraphics::IndexType}, \texttt{CoreGraphics::PrimitiveTopology}, \texttt{Frame::LightingMode},
Frame::SortingMode
- ToUpper() : Util::String
- transform() : Math::bbox
- transform44() : Math::transform44
- TransformDevice() : CoreGraphics::TransformDevice
- TransformDeviceBase() : Base::TransformDeviceBase
- TransformNode() : Models::TransformNode
- TransformNodeInstance() : Models::TransformNodeInstance
- TranslateKeyCode() : Win32::Win32DisplayDevice
- Trim() : Util::String
- TrimLeft() : Util::String
- TrimRight() : Util::String
- Type : Util::Variant, Base::ShaderVariableBase, Graphics::GraphicsEntity, Base::TextureBase, Input::InputEvent
- TypeToString() : Base::ShaderVariableBase, Util::Variant

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **u** -

  - Unload() : `Base::MeshBase`, `Base::VertexBufferBase`, `CoreGraphics::CPUVertexBuffer`, `Direct3D9::D3D9VertexBuffer`, `Models::Model`, `Direct3D9::D3D9IndexBuffer`, `Resources::Resource`, `CoreGraphics::CPUIndexBuffer`, `Direct3D9::D3D9Shader`, `Direct3D9::D3D9Texture`
  - UnloadResources() : `Models::StateNode`, `Models::Model`, `Models::ModelNode`, `Models::CharacterNode`, `Models::ParticleSystemNode`, `Models::ShapeNode`
  - Unmap() : `IO::FileStream`, `IO::MemoryStream`, `IO::Stream`, `IO::ZipFileStream`, `Base::IndexBufferBase`, `Base::TextureBase`, `Base::VertexBufferBase`, `CoreGraphics::CPUIndexBuffer`, `CoreGraphics::CPUVertexBuffer`, `Direct3D9::D3D9IndexBuffer`, `Direct3D9::D3D9Texture`, `Direct3D9::D3D9VertexBuffer`
  - UnmapCubeFace() : `Base::TextureBase`, `Direct3D9::D3D9Texture`
  - Unmount() : `IO::ZipFileSystem`
  - UnmountZipArchive() : `IO::IoServer`
  - Unregister() : `Attr::AttributeDefinitionBase`
  - UnregisterCommand() : `Scripting::LuaServer`, `Scripting::ScriptServer`
  - UnregisterSharedResource() : `Resources::SharedResourceServer`
  - UnregisterUriScheme() : `IO::IoServer`
- `Update()` : `Models::ModelNodeInstance`, `IO::Console`, `IO::ConsoleHandler`, `Models::ModelInstance`, `Models::CharacterNodeInstance`, `Models::ParticleSystemNodeInstance`, `Models::SkinShapeNodeInstance`, `RenderUtil::MayaCameraUtil`, `Resources::ResourceManager`, `Models::TransformNodeInstance`  
- `UpdateButtonState()` : `XInput::XInputGamePad`  
- `UpdateCameraLinks()` : `Graphics::Stage`  
- `UpdateEntities()` : `Graphics::Stage`  
- `UpdateGlobalBoundingBox()` : `Graphics::GraphicsEntity`  
- `UpdateLightLinks()` : `Graphics::Stage`  
- `UpdateLinks()` : `Graphics::Cell`  
- `UpdateManagedTextureVariables()` : `Models::StateNode`  
- `UpdatePositionInCellTree()` : `Graphics::GraphicsEntity`  
- `UpdateProgressDisplay()` : `BaseGameFeature::LoaderServer`  
- `UpdateRenderStats()` : `Resources::ManagedResource`  
- `UpdateShadowBuffers()` : `Lighting::SM30ShadowServer`  
- `UpdateThumbAxis()` : `XInput::XInputGamePad`  
- `UpdateTriggerAxis()` : `XInput::XInputGamePad`  
- `UpdateTriggeredEntities()` : `BaseGameFeature::EntityManager`  
- `UpdateViewProjMatrix()` : `AsyncGraphics::CameraEntityProxy`, `Graphics::CameraEntity`  
- `UpdateVisibilityLinks()` : `Graphics::View`  
- `upvec()` : `Math::vector`  
- `URI()` : `IO::URI`  
- `Usage` : `Base::ResourceBase`  
- `UserInfo()` : `IO::URI`  
- `UserProfile()` : `BaseGameFeature::UserProfile`  
- `UTF8toANSI()` : `Util::String`
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- v -

- ValidateCharacter(): `Models::CharacterNodeInstance`, `Models::SkinShapeNodeInstance`
- ValidateEntityHandle(): `AsyncGraphics::GraphicsEntityProxy`
- ValidateModelInstance(): `Graphics::ModelEntity`
- Value(): `Util::SimpleTree< VALUETYPE >::Node`, `Util::Atom< TYPE >`, `Util::KeyValuePair< KEYTYPE, VALUETYPE >`
- ValueAsString(): `Attr::Attribute`
- ValueAtIndex(): `Util::Dictionary< KEYTYPE, VALUETYPE >`
- Values(): `BaseGameFeature::CategoryManager::Entry`
- ValuesAsArray(): `Util::Dictionary< KEYTYPE, VALUETYPE >`
- ValueTypeToString(): `Attr::Attribute`
- Variant(): `Util::Variant`
- vec(): `Math::line`
- vector(): `Math::vector`
- VertexBufferBase(): `Base::VertexBufferBase`
- VertexComponent(): `CoreGraphics::VertexComponent`
- VertexLayoutBase(): `Base::VertexLayoutBase`
- VertexLayoutServer(): `CoreGraphics::VertexLayoutServer`
- View(): `Graphics::View`
- ViewerApplication(): `App::ViewerApplication`
- ViewProxy(): `AsyncGraphics::ViewProxy`
- VisResolveContainer(): `Models::VisResolveContainer< TYPE >`
- VisResolver(): `Models::VisResolver`
The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- W -

- Wait() : **Messaging::AsyncPort**, **Threading::SafePriorityQueue**< PRITYPE, TYPE >, **Win32::Win32Event**, **Threading::SafeQueue**< TYPE >
- WaitTimeout() : **Win32::Win32Event**
- Warning() : **IO::ConsoleHandler**, **Win32::Win32ConsoleHandler**, **IO::Console**
- Weekday : **Base::CalendarTimeBase**
- WeekdayToString() : **Base::CalendarTimeBase**
- WheelBackward() : **Base::MouseBase**
- WheelForward() : **Base::MouseBase**
- width() : **Math::rectangle**< TYPE >
- Width() : **Util::FixedTable**< TYPE >
- Win32ConsoleHandler() : **Win32::Win32ConsoleHandler**
- Win32CriticalSection() : **Win32::Win32CriticalSection**
- Win32DisplayDevice() : **Win32::Win32DisplayDevice**
- Win32Event() : **Win32::Win32Event**
- Win32FileTime() : **Win32::Win32FileTime**
- Win32Guid() : **Win32::Win32Guid**
- Win32Heap() : **Win32::Win32Heap**
- Win32InputServer() : **Win32::Win32InputServer**
- Win32IpAddress() : **Win32::Win32IpAddress**
- Win32Socket() : **Win32::Win32Socket**
- Win32Thread() : **Win32::Win32Thread**
- Win32Timer() : **Win32::Win32Timer**
- WinProc() : **Win32::Win32DisplayDevice**
- Write() : **Http::Base64Writer**, **IO::FileStream**,
IO::MemoryStream, IO::Stream, Win32::Win32FSWrapper, System::Win32Registry

- `WriteBlob()` : IO::BinaryWriter
- `WriteBool()` : IO::BinaryWriter
- `WriteChar()` : IO::BinaryWriter, IO::TextWriter
- `WriteContent()` : IO::XmlWriter
- `WriteDouble()` : IO::BinaryWriter
- `WriteFloat()` : IO::BinaryWriter
- `WriteFloat4()` : 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
- `WriteModelAttrs()` : Models::XmlModelWriter, Models::BinaryModelWriter, Models::ModelWriter
- `WriteModelNodeAttrs()` : Models::ModelWriter, Models::BinaryModelWriter, Models::XmlModelWriter
- `WriteResponse()` : Http::HttpResponseWriter
- `WriteShort()` : IO::BinaryWriter
- `WriteString()` : IO::TextWriter, IO::BinaryWriter
- `WriteUChar()` : IO::BinaryWriter
- `WriteUInt()` : IO::BinaryWriter
- `WriteUShort()` : IO::BinaryWriter
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
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
  - XInputGamePad() : XInput::XInputGamePad
  - XmlModelReader() : Models::XmlModelReader
  - XmlModelWriter() : Models::XmlModelWriter
  - XmlReader() : IO::XmlReader
  - XmlWriter() : IO::XmlWriter
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
- Year : Base::CalendarTimeBase
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- **Z** -

- 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
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- ~

- ~AbstractLightEntity() : \texttt{Lighting::AbstractLightEntity}
- ~ActorEntity() : \texttt{Graphics::ActorEntity}
- ~Application() : \texttt{App::Application}
- ~ArgsBlock() : \texttt{Scripting::ArgsBlock}
- ~Array() : \texttt{Util::Array< TYPE >}
- ~AsyncGraphicsHandler() : \texttt{AsyncGraphics::AsyncGraphicsHandler}
- ~AsyncGraphicsInterface() : \texttt{AsyncGraphics::AsyncGraphicsInterface}
- ~AsyncHttpInterface() : \texttt{AsyncHttp::AsyncHttpInterface}
- ~AsyncPort() : \texttt{Messaging::AsyncPort}
- ~AsyncRenderApplication() : \texttt{App::AsyncRenderApplication}
- ~AsyncViewerApplication() : \texttt{App::AsyncViewerApplication}
- ~Atom() : \texttt{Util::Atom< TYPE >}
- ~AttributeContainer() : \texttt{Attr::AttributeContainer}
- ~AttributeDefinitionBase() : \texttt{Attr::AttributeDefinitionBase}
- ~AttributeTable() : \texttt{Attr::AttributeTable}
- ~Base64Writer() : \texttt{Http::Base64Writer}
- ~BinaryModelReader() : \texttt{Models::BinaryModelReader}
- ~BinaryModelWriter() : \texttt{Models::BinaryModelWriter}
- ~BinaryReader() : \texttt{IO::BinaryReader}
- ~BinaryWriter() : \texttt{IO::BinaryWriter}
- ~Blob() : \texttt{Util::Blob}
- ~CameraEntity() : \texttt{Graphics::CameraEntity}
- ~CameraEntityProxy() : \texttt{AsyncGraphics::CameraEntityProxy}
- ~CategoryManager() : \texttt{BaseGameFeature::CategoryManager}
~Cell(): Graphics::Cell
~CharacterNode(): Models::CharacterNode
~CharacterNodeInstance(): Models::CharacterNodeInstance
~Command(): Scripting::Command
~Console(): IO::Console
~ConsoleApplication(): App::ConsoleApplication
~ConsoleHandler(): IO::ConsoleHandler
~CoreServer(): Core::CoreServer
~CPUIndexBuffer(): CoreGraphics::CPUIndexBuffer
~CPUVertexBuffer(): CoreGraphics::CPUVertexBuffer
~D3D9DisplayDevice(): Direct3D9::D3D9DisplayDevice
~D3D9IndexBuffer(): Direct3D9::D3D9IndexBuffer
~D3D9RenderDevice(): Direct3D9::D3D9RenderDevice
~D3D9RenderTarget(): Direct3D9::D3D9RenderTarget
~D3D9Shader(): Direct3D9::D3D9Shader
~D3D9ShaderInstance(): Direct3D9::D3D9ShaderInstance
~D3D9ShaderServer(): Direct3D9::D3D9ShaderServer
~D3D9ShaderVariable(): Direct3D9::D3D9ShaderVariable
~D3D9ShaderVariation(): Direct3D9::D3D9ShaderVariation
~D3D9ShapeRenderer(): Direct3D9::D3D9ShapeRenderer
~D3D9Texture(): Direct3D9::D3D9Texture
~D3D9VertexBuffer(): Direct3D9::D3D9VertexBuffer
~D3D9VertexLayout(): Direct3D9::D3D9VertexLayout
~DisplayDevice(): CoreGraphics::DisplayDevice
~DisplayDeviceBase(): Base::DisplayDeviceBase
~DisplayEventHandler(): CoreGraphics::DisplayEventHandler
~DisplayProxy(): AsyncGraphics::DisplayProxy
~DynamicMeshResourceLoader(): Resources::DynamicMeshResourceLoader
~Entity(): Game::Entity
~EntityLoaderBase(): BaseGameFeature::EntityLoaderBase
~EntityManager(): BaseGameFeature::EntityManager
~EnvEntityManager(): BaseGameFeature::EnvEntityManager
~EnvQueryManager(): BaseGameFeature::EnvQueryManager
~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`
- `~FramePostEffect()` : `Frame::FramePostEffect`
- `~FrameServer()` : `Frame::FrameServer`
- `~FrameShader()` : `Frame::FrameShader`
- `~GameApplication()` : `App::GameApplication`
- `~GamePadBase()` : `Base::GamePadBase`
- `~GameServer()` : `Game::GameServer`
- `~GameStateHandler()` : `App::GameStateHandler`
- `~GlobalAttrsManager()` : `BaseGameFeature::GlobalAttrsManager`
- `~GraphicsEntity()` : `Graphics::GraphicsEntity`
- `~GraphicsServer()` : `Graphics::GraphicsServer`
- `~Handler()` : `Messaging::Handler`
- `~HtmlPageWriter()` : `Http::HtmlPageWriter`
- `~HttpRequest()` : `Http::HttpRequest`
- `~HttpRequestHandler()` : `Http::HttpRequestHandler`
- `~HttpServer()` : `Http::HttpServer`
- `~IndexBufferBase()` : `Base::IndexBufferBase`
- `~InputHandler()` : `Input::InputHandler`
- `~InputServer()` : `Input::InputServer`
- `~InputServerBase()` : `Base::InputServerBase`
- `~IoServer()` : `IO::IoServer`
- `~KeyboardBase()` : `Base::KeyboardBase`
- `~LightServer()` : `Lighting::LightServer`
- `~LightServerBase()` : `Lighting::LightServerBase`
- `~List()` : `Util::List<TYPE>`
- `~LoaderServer()` : `BaseGameFeature::LoaderServer`
- `~LuaServer()` : `Scripting::LuaServer`
- `~ManagedResource()` : `Resources::ManagedResource`
- `~Manager()` : `Game::Manager`
- `~MemoryStream()` : `IO::MemoryStream`
- `~MeshBase()` : `Base::MeshBase`
- `~Model()` : `Models::Model`
- `~ModelEntity()` : `Graphics::ModelEntity`
- ModelInstance() : Models::ModelInstance
- ModelNode() : Models::ModelNode
- ModelNodeInstance() : Models::ModelNodeInstance
- ModelReader() : Models::ModelReader
- ModelServer() : Models::ModelServer
- ModelWriter() : Models::ModelWriter
- MouseBase() : Base::MouseBase
- MouseGripperProperty() :
  PhysicsFeature::MouseGripperProperty
- N2ModelReader() : Models::N2ModelReader
- Node() : Util::QuadTree< TYPE >::Node, Util::SimpleTree< VALUETYPE >::Node
- ParticleSystemNode() : Models::ParticleSystemNode
- ParticleSystemNodeInstance() :
  Models::ParticleSystemNodeInstance
- PhysicsProperty() : PhysicsFeature::PhysicsProperty
- Property() : Game::Property
- QuadtreeStageBuilder() : Graphics::QuadtreeStageBuilder
- RefCounted() : Core::RefCounted
- RenderApplication() : App::RenderApplication
- RenderDevice() : CoreGraphics::RenderDevice
- RenderDeviceBase() : Base::RenderDeviceBase
- RenderEventHandler() : CoreGraphics::RenderEventHandler
- RenderTargetBase() : Base::RenderTargetBase
- Resource() : Resources::Resource
- ResourceBase() : Base::ResourceBase
- ResourceLoader() : Resources::ResourceLoader
- ResourceManager() : Resources::ResourceManager
- ResourceMapper() : Resources::ResourceManager
- ResourceSaver() : Resources::ResourceSaver
- ScriptServer() : Scripting::ScriptServer
- 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`
- `~SharedResourceServer()` : `Resources::SharedResourceServer`
- `~SimpleResourceMapper()` : `Resources::SimpleResourceMapper`
- `~SimpleStageBuilder()` : `Graphics::SimpleStageBuilder`
- `~SkinShapeNode()` : `Models::SkinShapeNode`
- `~SkinShapeNodeInstance()` : `Models::SkinShapeNodeInstance`
- `~SM30LightServer()` : `Lighting::SM30LightServer`
- `~SM30ShadowServer()` : `Lighting::SM30ShadowServer`
- `~Stage()` : `Graphics::Stage`
- `~StageBuilder()` : `Graphics::StageBuilder`
- `~StateNode()` : `Models::StateNode`
- `~StateNodeInstance()` : `Models::StateNodeInstance`
- `~Stream()` : `IO::Stream`
- `~StreamAnimationLoader()` : `CoreGraphics::StreamAnimationLoader`
- `~StreamMeshLoader()` : `CoreGraphics::StreamMeshLoader`
- `~StreamModelLoader()` : `Models::StreamModelLoader`
- `~StreamReader()` : `IO::StreamReader`
- `~StreamTextureSaverBase()` : `Base::StreamTextureSaverBase`
- `~StreamWriter()` : `IO::StreamWriter`
- `~String()` : `Util::String`
- `~TcpClient()` : `Net::TcpClient`
- `~TcpClientConnection()` : `Net::TcpClientConnection`
- `~TcpServer()` : `Net::TcpServer`
- `~TextureBase()` : `Base::TextureBase`
- `~ThreadSafeDisplayEventHandler()` : `CoreGraphics::ThreadSafeDisplayEventHandler`
- `~ThreadSafeRenderEventHandler()` : `CoreGraphics::ThreadSafeRenderEventHandler`
- `~TransformDevice()` : `CoreGraphics::TransformDevice`
- `~TransformDeviceBase()` : `Base::TransformDeviceBase`
- `~TransformNode()` : `Models::TransformNode`
- `~TransformNodeInstance()` : `Models::TransformNodeInstance`
- Alloc() : Memory
- Clear() : Memory
- Copy() : Memory
- DuplicateCString() : Memory
- Exit() : Application
- Free() : Memory
- 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_lerp() : Math
- n_log2() : Math
- n_modangle() : Math
- n_rand() : Math
- n_saturate() : Math
- n_smooth() : Math
- Realloc() : Memory
- Sleep() : Timing
- n_barf() : debug.h
- n_barf2() : debug.h
- n_sleep() : debug.h
- operator delete() : win32memory.h
- operator delete[]() : win32memory.h
- operator new() : win32memory.h
- operator new[]() : win32memory.h
Main Page
Namespaces
Data Structures
Files
Related Pages

File List
Globals

All
Functions
Defines

__attribute___ : config.h
DEFAULT_IO_SCHEME : config.h
NEBULA3_MEMORY_STATS : config.h

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:54 2008
- b -

- Back() : Util::Array< TYPE >, Util::List< TYPE >, Util::SimpleTree< VALUETYPE >::Node
- Base64Writer() : Http::Base64Writer
- bbox() : Math::bbox
- Begin() : Http::HtmlPageWriter, Util::Array< TYPE >, Util::Crc, Util::FixedArray< TYPE >, Util::List< TYPE >, Base::ShaderInstanceBase, Base::ShapeRendererBase, Direct3D9::D3D9ShaderInstance
- begin_extend() : Math::bbox
- BeginAddCategoryAttrs() : BaseGameFeature::CategoryManager
- BeginAddColumns() : Attr::AttributeTable
- BeginAttachVisibleLights() : Lighting::LightServerBase
- BeginBatch() : Base::RenderDeviceBase, Base::RenderTargetBase
- BeginCapture() : Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- BeginClips() : Models::CharacterNode
- BeginFragments() : Models::SkinShapeNode
- BeginFrame() : Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice, Base::InputServerBase, Lighting::LightServerBase
- BeginJointPalette() : Models::SkinShapeNode
- BeginJoints() : Models::CharacterNode
- BeginModel() : Models::BinaryModelWriter, Models::ModelWriter, Models::XmlModelWriter
- BeginModelNode() : Models::BinaryModelWriter,
Models::ModelWriter, Models::XmlModelWriter
- BeginNode(): IO::XmlWriter
- BeginPass(): Direct3D9::D3D9RenderTarget, Base::RenderDeviceBase, Base::RenderTargetBase, Base::ShaderInstanceBase, Direct3D9::D3D9ShaderInstance
- BeginResolve(): Models::VisResolver
- BeginVariations(): Models::CharacterNode
- BinaryModelReader(): Models::BinaryModelReader
- BinaryModelWriter(): Models::BinaryModelWriter
- BinaryReader(): IO::BinaryReader
- BinarySearchIndex(): Util::Array< TYPE >, Util::FixedArray< TYPE >
- BinaryWriter(): IO::BinaryWriter
- Bind(): Win32::Win32Socket
- Blob(): Util::Blob
- BlobAttrId(): Attr::BlobAttrId
- BoolAttrId(): Attr::BoolAttrId
- BuildSignature(): Base::VertexLayoutBase
- BuildStage(): Graphics::SimpleStageBuilder, Graphics::QuadtreeStageBuilder, Graphics::StageBuilder
- ButtonDoubleClicked(): Base::MouseBase
- ButtonDown(): Base::MouseBase, Base::GamePadBase
- ButtonPressed(): Base::MouseBase, Base::GamePadBase
- ButtonUp(): Base::MouseBase, Base::GamePadBase
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
CalendarTimeBase() : Base::CalendarTimeBase
CameraDistance() : GraphicsFeature::CameraDistance
CameraEntity() : Graphics::CameraEntity
CameraEntityProxy() : AsyncGraphics::CameraEntityProxy
CameraFocus() : GraphicsFeature::CameraFocus
CameraOrbit() : GraphicsFeature::CameraOrbit
CanBeMapped() : IO::MemoryStream, IO::Stream, IO::ZipFileStream, IO::FileStream
Cancel() : Messaging::AsyncPort
CanCreate() : Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
CanLoadAsync() : CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader,
Models::StreamModelLoader, Resources::ResourceLoader,
Direct3D9::D3D9StreamShaderLoader,
Direct3D9::D3D9StreamTextureLoader
CanRead() : IO::FileStream, IO::MemoryStream, IO::Stream, IO::ZipFileStream
CanSeek() : IO::FileStream, IO::MemoryStream, IO::Stream, IO::ZipFileStream
CanWrite() : IO::FileStream, IO::MemoryStream, IO::Stream, IO::ZipFileStream
Capacity() : Util::Array< TYPE >, Util::HashTable< KEYTYPE, VALUETYPE >
Category() : BaseGameFeature::CategoryManager::Category, BaseGameFeature::CategoryManager::Entry
CategoryManager() : BaseGameFeature::CategoryManager
- Cell() : Graphics::Cell
- center() : Math::bbox
- CharacterNode() : Models::CharacterNode
- CharacterNodeInstance() : Models::CharacterNodeInstance
- CheckFileExtension() : Util::String
- CheckId() : Messaging::Message
- CheckValidCharSet() : Util::String
- Child() : Util::SimpleTree< VALUETYPE >::Node
- ClassExists() : Core::Factory
- Cleanup() : Base::ShaderInstanceBase, Direct3D9::D3D9ShaderInstance, BaseGameFeature::EntityManager
- CleanupGameFeatures() : App::GameApplication
- CleanupStateHandlers() : App::GameApplication
- Clear() : Util::FixedArray< TYPE >, Util::FixedTable< TYPE >, Util::HashTable< KEYTYPE, VALUETYPE >, Util::List< TYPE >, Util::Proxy< TYPE >, Util::Queue< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::Stack< TYPE >, Util::String, Util::Variant, Resources::ManagedResource, Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::MediaType, IO::URI, Scripting::ArgsBlock, Threading::SafePriorityQueue< PRITYPE, TYPE >, Threading::SafeQueue< TYPE >, Util::Array< TYPE >, Util::Atom< TYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >
- ClearAssign() : IO::IoServer
- ClearCapture() : Base::InputServerBase
- ClearDataPtr() : Util::QuadTree< TYPE >::Node
- ClearDeletedRowsFlags() : Attr::AttributeTable
- ClearDynamicAttributes() : Attr::AttributeDefinitionBase
- ClearEntity() : Game::Property
- ClearEnvEntity() : BaseGameFeature::EnvEntityManager
- ClearError() : Scripting::ScriptServer
- ClearFeatureBits() : Base::ShaderServerBase
- ClearKeyboardCapture() : Base::InputServerBase
- ClearLinks() : Graphics::GraphicsEntity
- ClearMouseCapture() : Base::InputServerBase
- ClearNewRowFlags() : Attr::AttributeTable
- ClearRenderStats() : Resources::ManagedResource
- `clipstatus()`: `Math::bbox`, `Math::sphere`
- `Close()`: `Direct3D9::D3D9RenderDevice`, `Direct3D9::D3D9ShaderServer`, `Direct3D9::D3D9ShapeRenderer`, `CoreGraphics::VertexLayoutServer`, `Win32::Win32DisplayDevice`, `Frame::FrameServer`, `Graphics::GraphicsServer`, `Base::InputServerBase`, `Win32::Win32InputServer`, `Lighting::LightServerBase`, `Lighting::SM30LightServer`, `Lighting::SM30ShadowServer`, `Models::BinaryModelReader`, `Models::ModelReader`, `Models::N2ModelReader`, `Models::XmlModelReader`, `Models::ModelServer`, `Models::BinaryModelWriter`, `Models::ModelWriter`, `Models::XmlModelWriter`, `Models::VisResolver`, `Resources::ResourceManager`, `Resources::SharedResourceServer`, `App::GameApplication`, `BaseGameFeature::LoaderServer`, `Game::GameServer`, `App::Application`, `App::ConsoleApplication`, `Core::CoreServer`, `Http::HtmlPageWriter`, `Http::HttpServer`, `IO::BinaryReader`, `IO::BinaryWriter`, `IO::Console`, `IO::ConsoleHandler`, `IO::FileStream`, `IO::MemoryStream`, `IO::Stream`, `IO::StreamReader`, `IO::StreamWriter`, `IO::XmlReader`, `IO::XmlWriter`, `IO::ZipFileStream`, `IO::ZipArchive`, `Messaging::AsyncPort`, `Messaging::Handler`, `Base::TransformDeviceBase`, `Net::TcpServer`, `Win32::Win32Socket`, `Scripting::LuaServer`, `Base::DisplayDeviceBase`, `Scripting::ScriptServer`, `App::AsyncRenderApplication`, `App::AsyncViewerApplication`, `App::RenderApplication`, `App::ViewerApplication`, `AsyncGraphics::AsyncGraphicsHandler`, `AsyncGraphics::DisplayProxy`, `AsyncGraphics::GraphicsServerProxy`, `Base::RenderDeviceBase`, `Base::ShaderServerBase`, `Base::ShapeRendererBase`
- `CloseDInputMouse()`: `Win32::Win32InputServer`
- `CloseFile()`: `Win32::Win32FSWrapper`
- `CloseProgressIndicator()`: `BaseGameFeature::LoaderServer`
- `closestpoint()`: `Math::line`
- `CloseWindow()`: `Win32::Win32DisplayDevice`
- CmdLineArgs() : Util::CmdLineArgs
- Column() : Util::QuadTree< TYPE >::Node
- Command() : Scripting::Command
- Commit() : Base::ShaderInstanceBase, Direct3D9::D3D9ShaderInstance
- CommitChangesToDatabase() : BaseGameFeature::CategoryManager
- Compute() : Util::Crc, Lighting::PSSMUtil
- ComputeAbsMousePos() : Win32::Win32DisplayDevice
- ComputeAdjustedWindowRect() : Win32::Win32DisplayDevice
- ComputeClipStatus() : Graphics::CameraEntity, Graphics::GraphicsEntity, Lighting::GlobalLightEntity, Lighting::SpotLightEntity
- ComputeFileCrc() : IO::IoServer
- ComputeMouseWorldRay() : BaseGameFeature::EnvQueryManager
- ComputeNormMousePos() : Win32::Win32DisplayDevice
- ComputeWorldMouseRay() : Graphics::GraphicsServer
- Concatenate() : Util::String
- Connect() : Win32::Win32Socket, Net::TcpClientConnection, Net::TcpClient
- Console() : IO::Console
- ConsoleApplication() : App::ConsoleApplication
- ConsoleHandler() : IO::ConsoleHandler
- contains() : Math::bbox
- Contains() : Util::Queue< TYPE >, Util::HashTable< KEYTYPE, VALUETYPE >
- contains() : Math::bbox
- Contains() : Util::Dictionary< KEYTYPE, VALUETYPE >, Util::Stack< TYPE >
- ContainsCharFromSet() : Util::String
- Content() : Util::HashTable< KEYTYPE, VALUETYPE >
- ConvertBackslashes() : Util::String
- ConvertDirToZipURIIfExists() : IO::ZipFileSystem
- ConvertDouble() : System::ByteOrder
- ConvertFileToZipURIIfExists() : IO::ZipFileSystem
- ConvertFloat() : System::ByteOrder
- ConvertFloat4() : System::ByteOrder
- ConvertInt() : System::ByteOrder
- ConvertMatrix44() : System::ByteOrder
- ConvertShort() : System::ByteOrder
- ConvertToPathInZipArchive() : IO::ZipArchive
- ConvertToZipURI() : IO::ZipArchive
- ConvertUInt() : System::ByteOrder
- ConvertUShort() : System::ByteOrder
- CopyExtRow() : Attr::AttributeTable
- CopyFile() : IO::IoServer, Interface::CopyFile
- CopyRow() : Attr::AttributeTable
- CorePageHandler() : Debug::CorePageHandler
- CoreServer() : Core::CoreServer
- corner_point() : Math::bbox
- CPUIndexBuffer() : CoreGraphics::CPUIndexBuffer
- CPUVertexBuffer() : CoreGraphics::CPUVertexBuffer
- Crc() : Util::Crc
- Create() : Core::Factory, Core::Rtti, Core::Factory
- CreateClone() : Graphics::GraphicsEntity
- CreateDirectory() : Win32::Win32FSWrapper, 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
- CreateEnvEntity() : BaseGameFeature::EnvEntityManager
- 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::StateNode, Models::SkinShapeNode, Models::ModelNode, Models::ParticleSystemNode, Models::ShapeNode, Models::CharacterNode, Models::TransformNode
  CreateProperty(): BaseGameFeature::FactoryManager
  CreateShaderInstance(): Base::ShaderBase, Base::ShaderServerBase
  CreateShaderVariableInstance(): Models::StateNodeInstance
  CreateSharedResource(): Resources::SharedResourceServer
  CreateSharedVertexLayout(): CoreGraphics::VertexLayoutServer
  CreateStage(): Graphics::GraphicsServer
  CreateStageProxy(): AsyncGraphics::GraphicsServerProxy
  CreateStream(): IO::IoServer
  CreateUserProfile(): BaseGameFeature::LoaderServer
  CreateView(): Graphics::GraphicsServer
  CreateViewProxy(): AsyncGraphics::GraphicsServerProxy
```
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
D3D9DisplayDevice() : Direct3D9::D3D9DisplayDevice
D3D9IndexBuffer() : Direct3D9::D3D9IndexBuffer
D3D9RenderDevice() : Direct3D9::D3D9RenderDevice
D3D9RenderTarget() : Direct3D9::D3D9RenderTarget
D3D9Shader() : Direct3D9::D3D9Shader
D3D9ShaderInstance() : Direct3D9::D3D9ShaderInstance
D3D9ShaderServer() : Direct3D9::D3D9ShaderServer
D3D9ShaderVariable() : Direct3D9::D3D9ShaderVariable
D3D9ShaderVariation() : Direct3D9::D3D9ShaderVariation
D3D9ShapeRenderer() : Direct3D9::D3D9ShapeRenderer
D3D9Texture() : Direct3D9::D3D9Texture
D3D9VertexBuffer() : Direct3D9::D3D9VertexBuffer
D3D9VertexLayout() : Direct3D9::D3D9VertexLayout
DeactivateEntity() : BaseGameFeature::EntityManager
DebugOut() : IO::Console, IO::ConsoleHandler, Win32::Win32ConsoleHandler, Win32::SysFunc
Decode() : Interface::CopyFile, Interface::IOMessage, Interface::ReadStream, Interface::WriteStream, Messaging::Message
DecrClientCount() : Resources::ManagedResource
Decrement() : Win32::Win32Interlocked
DecrUseCount() : Resources::Resource
DelayedJob() : BaseGameFeature::EntityManager::DelayedJob
Delete() : System::Win32Registry
DeleteAllRows() : Attr::AttributeTable
DeleteDirectory() : IO::IoServer, Win32::Win32FSWrapper
DeleteEntity() : BaseGameFeature::EntityManager
DeleteEntityImmediate() : BaseGameFeature::EntityManager
DeleteEnvEntity() : BaseGameFeature::EnvEntityManager
DeleteFile() : IO::IoServer, Win32::Win32FSWrapper
DeleteInstance() : BaseGameFeature::CategoryManager
DeleteLevel() : BaseGameFeature::CategoryManager
DeleteProfile() : BaseGameFeature::UserProfile
DeleteRow() : Attr::AttributeTable
Dequeue() : Threading::SafePriorityQueue< PRITYPE, TYPE >, Threading::SafeQueue< TYPE >, Util::Queue< TYPE >
DequeueAll() : Threading::SafeQueue< TYPE >
Destroy() : Attr::AttributeDefinitionBase, Core::Factory
diagonal_size() : Math::bbox
Dictionary() : Util::Dictionary< KEYTYPE, VALUETYPE >
Difference() : Util::Array< TYPE >
DirectoryExists() : IO::IoServer, Win32::Win32FSWrapper
DisablePhysics() : PhysicsFeature::PhysicsProperty
Discard() : Frame::FrameShader, Models::ModelInstance, Frame::FramePostEffect, Base::VertexLayoutBase, AsyncGraphics::GraphicsEntityProxy, Base::RenderTargetBase, Base::ShaderInstanceBase, Direct3D9::D3D9RenderTarget, Frame::FrameBatch, Frame::FramePass
DiscardAllStageProxies() : AsyncGraphics::GraphicsServerProxy
DiscardAllStages() : Graphics::GraphicsServer
DiscardAllViewProxies() : AsyncGraphics::GraphicsServerProxy
DiscardAllViews() : Graphics::GraphicsServer
DiscardManagedModel() : Models::ModelServer
DiscardManagedResource() : Resources::ResourceManager
DiscardShaderInstance() : Base::ShaderBase
DiscardStage() : Graphics::GraphicsServer
DiscardStageProxy() : AsyncGraphics::GraphicsServerProxy
DiscardView() : Graphics::GraphicsServer
DiscardViewProxy() : AsyncGraphics::GraphicsServerProxy
Disconnect() : Net::TcpClient
Dispatcher() : Messaging::Dispatcher
DisplayDevice() : CoreGraphics::DisplayDevice
- DisplayDeviceBase() : Base::DisplayDeviceBase
- DisplayEvent() : CoreGraphics::DisplayEvent
- DisplayEventHandler() : CoreGraphics::DisplayEventHandler
- DisplayMode() : CoreGraphics::DisplayMode
- DisplayPageHandler() : Debug::DisplayPageHandler
- DisplayProxy() : AsyncGraphics::DisplayProxy
- distance() : Math::line
- DoStateTransition() : App::GameApplication
- DoWork() : Win32::Win32Thread, Messaging::Handler, AsyncGraphics::AsyncGraphicsHandler
- Draw() : Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
- DrawIndexedPrimitives() : Direct3D9::D3D9ShapeRenderer, Base::ShapeRendererBase
- DrawPrimitives() : Base::ShapeRendererBase, Direct3D9::D3D9ShapeRenderer
- DrawShape() : Direct3D9::D3D9ShapeRenderer, Base::ShapeRendererBase
- DumpLeaks() : Core::RefCountedList
- DumpRefCountingLeaks() : Core::RefCounted
- DuplicateLevel() : BaseGameFeature::CategoryManager
- DynamicMeshResourceLoader() : Resources::DynamicMeshResourceLoader
- e -

- Element() : **Http::HtmlPageWriter**
- EmitWakeupSignal() : **Win32::Win32Thread**
- EnablePhysics() : **PhysicsFeature::PhysicsProperty**
- Encode() : **Interface::CopyFile**, **Messaging::Message**, **Interface::IOMessage**, **Interface::ReadStream**, **Interface::WriteStream**
- End() : **Base::ShapeRendererBase**, **Direct3D9::D3D9ShaderInstance**
- end() : **Math::line**
- End() : **Http::HtmlPageWriter**, **Util::Array< TYPE >**, **Util::Crc**, **Util::FixedArray< TYPE >**, **Util::List< TYPE >**, **Base::ShaderInstanceBase**
- end_extend() : **Math::bbox**
- EndAddCategoryAttrs() : **BaseGameFeature::CategoryManager**
- EndAddColumns() : **Attr::AttributeTable**
- EndAttachVisibleLights() : **Lighting::LightServerBase**
- EndBatch() : **Base::RenderDeviceBase**, **Base::RenderTargetBase**
- EndCapture() : **Base::KeyboardBase**, **Base::MouseBase**, **Input::InputHandler**
- EndClips() : **Models::CharacterNode**
- EndFragments() : **Models::SkinShapeNode**
- EndFrame() : **Lighting::LightServerBase**, **Base::RenderDeviceBase**, **Direct3D9::D3D9RenderDevice**, **Base::InputServerBase**
- EndJointPalette() : **Models::SkinShapeNode**
- EndJoints() : **Models::CharacterNode**
- EndModel() : Models::BinaryModelWriter, Models::ModelWriter, Models::XmlModelWriter
- EndModelNode() : Models::BinaryModelWriter, Models::ModelWriter, Models::XmlModelWriter
- EndNode() : IO::XmlWriter
- EndPass() : Base::RenderDeviceBase, Base::RenderTargetBase, Base::ShaderInstanceBase, Direct3D9::D3D9RenderDevice, Direct3D9::D3D9RenderTarget, Direct3D9::D3D9ShaderInstance
- EndResolve() : Models::VisResolver
- EndVariations() : Models::CharacterNode
- Enqueue() : Threading::SafeQueue< TYPE >, Util::Queue< TYPE >
- Enter() : Win32::Win32CriticalSection
- Entity() : Game::Entity
- EntityIsInActiveLayer() : BaseGameFeature::EntityLoaderBase
- EntityLoaderBase() : BaseGameFeature::EntityLoaderBase
- EntityManager() : BaseGameFeature::EntityManager
- Entry() : BaseGameFeature::CategoryManager::Entry
- EnumProfiles() : BaseGameFeature::UserProfile
- EnvEntityExists() : BaseGameFeature::EnvEntityManager
- EnvEntityManager() : BaseGameFeature::EnvEntityManager
- EnvQueryManager() : BaseGameFeature::EnvQueryManager
- Eof() : IO::MemoryStream, IO::Stream, IO::StreamReader, IO::ZipFileStream, Win32::Win32FSWrapper, IO::FileStream
- Erase() : Util::Array< TYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::HashTable< KEYTYPE, VALUETYPE >
- EraseAtIndex() : Util::Dictionary< KEYTYPE, VALUETYPE >
- EraseIndex() : Util::Array< TYPE >
- EraseMatchingElements() : Threading::SafeQueue< TYPE >, Threading::SafePriorityQueue< PRITYPE, TYPE >
- Error() : IO::ConsoleHandler, Win32::Win32ConsoleHandler, Win32::SysFunc, IO::Console
- Eval() : Scripting::ScriptServer, Scripting::LuaServer
- EvalScript() : Scripting::ScriptServer
- EvaluateSkeleton() : Graphics::ActorEntity
- EvaluateVariation() : Models::CharacterNode
- Exists() : `System::Win32Registry`
- ExistsEntityByAttr() : `BaseGameFeature::EntityManager`
- ExistsEntityByUniqueId() : `BaseGameFeature::EntityManager`
- Exit() : `App::Application`, `Win32::SysFunc`
- extend() : `Math::bbox`
- extents() : `Math::bbox`
- ExtractDirName() : `Util::String`
- ExtractFileName() : `Util::String`
- ExtractFromUri() : `Win32::Win32IpAddress`
- ExtractLastDirName() : `Util::String`
- ExtractRange() : `Util::String`
- ExtractToEnd() : `Util::String`
- ExtractToLastSlash() : `Util::String`
FactoryManager() : BaseGameFeature::FactoryManager
EaseAnimation() : Graphics::ActorEntity
EaseAnimationMix() : Graphics::ActorEntity
EaseRunningAnimationsOut() : Graphics::ActorEntity
FeatureMaskToString() : Base::ShaderServerBase
FeatureStringToMask() : Base::ShaderServerBase
FeatureUnit() : Game::FeatureUnit
FileExists() : IO::IoServer, Win32::Win32FSWrapper
FileStream() : IO::FileStream
FileTimeToLocalTime() : Base::CalendarTimeBase, Win32::Win32CalendarTime
FileTimeToSystemTime() : Win32::Win32CalendarTime, Base::CalendarTimeBase
Fill() : Util::Array< TYPE >, Util::FixedArray< TYPE >
FillModel() : Models::BinaryModelReader, Models::ModelReader, Models::N2ModelReader, Models::XmlModelReader
Find() : Util::Array< TYPE >, Util::FixedArray< TYPE >, Util::List< TYPE >, Util::SimpleTree< VALUETYPE >::Node
FindBluePrint() : BaseGameFeature::FactoryManager
FindByFourCC() : Attr::AttributeDefinitionBase
FindByName() : Attr::AttributeDefinitionBase
FindCharIndex() : Util::String
FindComponent() : Base::VertexLayoutBase
FindContainmentNode() : Util::QuadTree< TYPE >::Node
FindDirEntry() : IO::ZipArchive, IO::ZipDirEntry
FindFileEntry() : IO::ZipArchive, IO::ZipDirEntry
- FindIndex() : Util::Array< TYPE >, Util::Dictionary< KEYTYPE, VALUETYPE >, Util::FixedArray< TYPE >
- FindInstances() : BaseGameFeature::CategoryManager
- FindProperty() : Game::Entity
- FindRowIndexByAttr() : Attr::AttributeTable
- FindRowIndexByAttrs() : Attr::AttributeTable
- FindRowIndicesByAttr() : Attr::AttributeTable
- FindRowIndicesByAttrs() : Attr::AttributeTable
- FindStateHandlerByName() : App::GameApplication
- FindStringIndex() : Util::String
- FindTemplate() : BaseGameFeature::CategoryManager
- FindTemplateByAttr() : BaseGameFeature::CategoryManager
- FindZipArchive() : IO::ZipFileSystem
- FindZipArchiveWithDir() : IO::ZipFileSystem
- FindZipArchiveWithFile() : IO::ZipFileSystem
- FixedArray() : Util::FixedArray< TYPE >
- FixedTable() : Util::FixedTable< TYPE >
- float2() : Math::float2
- Float4AttrId() : Attr::Float4AttrId
- FloatAttrId() : Attr::FloatAttrId
- Flush() : IO::FileStream, IO::Stream, Win32::Win32FSWrapper, Messaging::AsyncPort
- FocusManager() : BaseGameFeature::FocusManager
- Format() : Base::CalendarTimeBase, Util::String
- FormatArgList() : Util::String
- FormatToString() : CoreGraphics::VertexComponent
- FourCC() : Util::FourCC
- Fragment() : IO::URI
- FrameBatch() : Frame::FrameBatch
- FramePass() : Frame::FramePass
- FramePostEffect() : Frame::FramePostEffect
- FrameServer() : Frame::FrameServer
- FrameShader() : Frame::FrameShader
- Free() : Win32::Win32Heap
- FromBinary() : Win32::Win32Guid
- FromBool() : Util::String
- FromFloat() : Util::String
- FromFloat4() : Util::String
- FromInt() : Util::String
FromMatrix44() : Util::String
FromMediaType() : CoreGraphics::ImageFileFormat
FromString() : CoreGraphics::AntiAliasQuality, Http::HttpStatus, Win32::Win32Guid, Input::Key, CoreGraphics::Adapter, Frame::LightingMode, CoreGraphics::BatchType, Frame::SortingMode, CoreGraphics::PixelFormat, CoreGraphics::ImageFileFormat, Http::HttpMethod, CoreGraphics::PrimitiveTopology, Input::MouseButton, Models::ModelNodeType, Util::FourCC, CoreGraphics::IndexType
Front() : Util::List< TYPE >, Util::Array< TYPE >, Util::SimpleTree< VALUETYPE >::Node

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
- g -

- GameApplication() : `App::GameApplication`
- GamePadBase() : `Base::GamePadBase`
- GameServer() : `Game::GameServer`
- GameStateHandler() : `App::GameStateHandler`
- ge() : `Math::float2`
- gen() : `Math::noise`
- Generate() : `Win32::Win32Guid`
- GenerateMipLevels() : `Base::RenderTargetBase`, `Direct3D9::D3D9RenderTarget`
- Get() : `Models::VisResolveContainer< TYPE >`
- get_cartesian() : `Math::polar`
- get_clipplanes() : `Math::bbox`
- GetAbsMousePos() : `CoreGraphics::DisplayEvent`, `Input::InputEvent`
- GetAcceptedMessages() : `Messaging::Port`
- GetAccess() : `Base::ResourceBase`
- GetAccessMode() : `Attr::Attribute`, `Attr::AttributeDefinitionBase`, `Attr::AttrId`, `IO::Stream`
- GetAccessPattern() : `IO::Stream`
- GetAccessType() : `CoreGraphics::VertexComponent`
- GetActiveShaderInstance() : `Base::ShaderServerBase`
- GetActiveVariation() : `Base::ShaderInstanceBase`
- GetActivityDistance() : `Models::ParticleSystemNode`
- GetAdapter() : `AsyncGraphics::DisplayProxy`, `Base::DisplayDeviceBase`
- GetAdapterInfo() : `AsyncGraphics::DisplayProxy`, `Base::DisplayDeviceBase`, `Direct3D9::D3D9DisplayDevice`
- GetAddress() : Win32::Win32Socket, Net::TcpServer
- GetAllAssigns() : IO::IoServer
- GetAllAttrIds() : Attr::AttrId
- GetAllRegisteredUriSchemes() : IO::IoServer
- GetAllShaderInstances() : Base::ShaderBase
- GetAllShaders() : Base::ShaderServerBase
- GetAnim() : Models::CharacterNode
- GetAnimationMapping() : Graphics::ActorEntity
- GetAnimClipScheduler() : Graphics::ActorEntity
- GetAntiAliasQuality() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase, Base::RenderTargetBase
- GetAppName() : App::Application, Core::CoreServer
- GetArgName() : Scripting::ArgsBlock
- GetArguments() : Scripting::Command
- GetArgValue() : Scripting::ArgsBlock
- GetAspect() : AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
- GetAspectRatio() : CoreGraphics::DisplayMode
- GetAssign() : IO::IoServer
- GetAtomTableSize() : Util::Atom<TYPE>
- GetAttr() : Attr::AttributeContainer, Models::Model, Models::ModelNode, Game::Entity
- GetAttributes() : Graphics::StageBuilder
- GetAttrId() : Attr::Attribute
- GetAttrs() : Models::ModelNode, Attr::AttributeContainer, IO::XmlReader, Models::Model
- GetAttrTable() : Game::Entity
- GetAttrTableRowIndex() : Game::Entity
- GetAvailableDisplayModes() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase, Direct3D9::D3D9DisplayDevice
- GetAxisValue() : Base::GamePadBase
- GetBackLightColor() : Lighting::GlobalLightEntity
- GetBaseAnimation() : Graphics::ActorEntity
- GetBaseAnimationDuration() : Graphics::ActorEntity
- GetBaseClip() : Graphics::ActorEntity
- GetBaseIndex() : CoreGraphics::PrimitiveGroup
- GetBaseVertex() : CoreGraphics::PrimitiveGroup
- GetBatchByIndex() : Frame::FramePass
- GetBillboardOrientation() : Models::ParticleSystemNode
- GetBinDirectory() : \texttt{Win32::Win32FSWrapper}
- GetBlob() : \texttt{Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::GlobalAttrsManager, Game::Entity}
- GetBlobArray() : \texttt{Util::Variant}
- GetBlobDefValue() : \texttt{Attr::AttrId}
- GetBlocking() : \texttt{Win32::Win32Socket}
- GetBool() : \texttt{Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlReader, Util::CommandLineArgs, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity}
- GetBoolArray() : \texttt{Util::Variant}
- GetBoolAtIndex() : \texttt{Util::CommandLineArgs}
- GetBoolDefValue() : \texttt{Attr::AttrId}
- GetBoundingBox() : \texttt{Util::QuadTree< TYPE >::Node, CoreGraphics::PrimitiveGroup, Graphics::Cell, Models::Model, Models::ModelNode}
- GetBroadcast() : \texttt{Win32::Win32Socket}
- GetByteSize() : \texttt{CoreGraphics::VertexComponent}
- GetCameraEntity() : \texttt{Graphics::View, Lighting::PSSMUtil}
- GetCameraFocusEntity() : \texttt{BaseGameFeature::FocusManager}
- GetCameraTransform() : \texttt{RenderUtil::MayaCameraUtil}
- GetCastShadows() : \texttt{Lighting::AbstractLightEntity}
- GetCategory() : \texttt{Game::Entity}
- GetCategoryByIndex() : \texttt{BaseGameFeature::CategoryManager}
- GetCategoryByName() : \texttt{BaseGameFeature::CategoryManager}
- GetCell() : \texttt{Graphics::GraphicsEntity}
- GetChar() : \texttt{CoreGraphics::DisplayEvent, Input::InputEvent}
- GetCharacter() : \texttt{Models::CharacterNodeInstance, Models::CharacterNode}
- GetCharacter3Set() : \texttt{Graphics::ActorEntity}
- GetCharacterNode() : \texttt{Graphics::ActorEntity}
- GetCharacterPointer() : \texttt{Graphics::ActorEntity}
- GetCharacterSet() : \texttt{Models::CharacterNode, Models::CharacterNodeInstance}
- GetCharInput() : \texttt{Base::KeyboardBase}
- GetChildCells() : \texttt{Graphics::Cell}
- GetChildren(): `Models::ModelNode`, `Models::ModelNodeInstance`
- GetClassFourCC(): `Core::RefCounted`
- GetClassName(): `Core::RefCounted`
- GetClassRtti(): `Core::Factory`
- GetClearColor(): `Base::RenderTargetBase`, `Frame::FramePass`
- GetClearDepth(): `Base::RenderTargetBase`, `Frame::FramePass`
- GetClearStencil(): `Base::RenderTargetBase`, `Frame::FramePass`
- GetClientAddress(): `Net::TcpClientConnection`
- GetClientCount(): `Resources::ManagedResource`
- GetClipAt(): `Models::CharacterNode`
- GetClipDuration(): `Models::CharacterNode`
- GetClipIndexByName(): `Models::CharacterNode`
- GetCmdLineArgs(): `App::Application`
- GetCmdName(): `Util::CmdLineArgs`
- GetColor(): `Lighting::AbstractLightEntity`
- GetColorBufferFormat(): `Base::RenderTargetBase`
- GetColumnAccessMode(): `Attr::AttributeTable`
- GetColumnFourCC(): `Attr::AttributeTable`
- GetColumnId(): `Attr::AttributeTable`
- GetColumnIndex(): `Attr::AttributeTable`
- GetColumnName(): `Attr::AttributeTable`
- GetColumnType(): `Attr::AttributeTable`
- GetCommandByIndex(): `Scripting::ScriptServer`
- GetCommandByName(): `Scripting::ScriptServer`
- GetCompanyName(): `App::Application`, `Core::CoreServer`
- GetComponentAt(): `Base::VertexLayoutBase`
- GetComponentByteOffset(): `Base::VertexLayoutBase`
- GetContent(): `IO::XmlReader`
- GetCoreId(): `Win32::Win32Thread`
- GetCurrentAdapterDisplayMode(): `AsyncGraphics::DisplayProxy`, `Base::DisplayDeviceBase`, `Direct3D9::D3D9DisplayDevice`
- GetCurrentNodeLineNumber(): `IO::XmlReader`
- GetCurrentNodeName(): `IO::XmlReader`
- GetCurrentNodePath(): `IO::XmlReader`
- GetCurrentState() : App::GameApplication
- GetCurrentStateHandler() : App::GameApplication
- GetCurve() : Models::ParticleSystemNode
- GetD3D9BaseTexture() : Direct3D9::D3D9Texture
- GetD3D9CubeTexture() : Direct3D9::D3D9Texture
- GetD3D9Effect() : Direct3D9::D3D9Shader, Direct3D9::D3D9ShaderInstance, Direct3D9::D3D9ShaderVariation
- GetD3D9EffectPool() : Direct3D9::D3D9ShaderServer
- GetD3D9IndexBuffer() : Direct3D9::D3D9IndexBuffer
- GetD3D9Technique() : Direct3D9::D3D9ShaderVariation
- GetD3D9Texture() : Direct3D9::D3D9Texture
- GetD3D9VertexBuffer() : Direct3D9::D3D9VertexBuffer
- GetD3D9VertexDeclaration() : Direct3D9::D3D9VertexLayout
- GetD3D9VolumeTexture() : Direct3D9::D3D9Texture
- GetDatabasePath() : BaseGameFeature::UserProfile
- GetDay() : Base::CalendarTimeBase
- GetDebugTextEnabled() : BaseGameFeature::LoaderServer
- GetDefaultGamePad() : Base::InputServerBase
- GetDefaultKeyboard() : Base::InputServerBase
- GetDefaultMouse() : Base::InputServerBase
- GetDefaultRenderTarget() : Base::RenderDeviceBase
- GetDefaultValue() : Attr::AttributeDefinitionBase
- GetDefaultView() : Graphics::GraphicsServer
- GetDefaultViewProxy() : AsyncGraphics::GraphicsServerProxy
- GetDeletedRowIndices() : Attr::AttributeTable
- GetDependencies() : Graphics::View
- GetDepth() : Base::TextureBase
- GetDesc() : Http::HttpRequestHandler
- GetDescription() : CoreGraphics::AdapterInfo
- GetDeviceId() : CoreGraphics::AdapterInfo
- GetDeviceName() : CoreGraphics::AdapterInfo
- GetDirect3D() : Direct3D9::D3D9RenderDevice
- GetDirect3DDevice() : Direct3D9::D3D9RenderDevice
- GetDirEntries() : IO::ZipDirEntry
- GetDisplayMode() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- GetDistance() : BaseGameFeature::MoveFollow
- GetDontLinger() : Win32::Win32Socket
GetDriverName() : CoreGraphics::AdapterInfo
GetDriverVersionHighPart() : CoreGraphics::AdapterInfo
GetDriverVersionLowPart() : CoreGraphics::AdapterInfo
GetEmissionDuration() : Models::ParticleSystemNode
GetEntities() : Graphics::Cell, Graphics::Stage, BaseGameFeature::EntityManager, GraphicsFeature::GetGraphicsEntities
GetEntitiesByAttr() : BaseGameFeature::EntityManager
GetEntitiesByAttrs() : BaseGameFeature::EntityManager
GetEntitiesByType() : Graphics::Cell, Graphics::Stage
GetEntitiesInActivityBubble() : BaseGameFeature::EntityManager
GetEntitiesInBox() : BaseGameFeature::EnvQueryManager
GetEntitiesInSphere() : BaseGameFeature::EnvQueryManager
GetEntity() : Game::Property
GetEntityByAttr() : BaseGameFeature::EntityManager
GetEntityByAttrs() : BaseGameFeature::EntityManager
GetEntityById() : BaseGameFeature::EntityManager
GetEntityUnderMouse() : BaseGameFeature::EnvQueryManager
GetError() : Scripting::Command, Scripting::ScriptServer
GetErrorCode() : Win32::Win32Socket
GetErrorString() : Win32::Win32Socket
GetEventCode() : CoreGraphics::DisplayEvent, CoreGraphics::RenderEvent
GetFarHeight() : AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
GetFarWidth() : AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
GetFeatureBits() : Base::ShaderServerBase
GetFeatureMask() : Base::ShaderVariationBase
GetFileEntries() : IO::ZipDirEntry
GetFileExtension() : Util::String, Models::BinaryModelWriter, Models::ModelWriter, Models::XmlModelWriter
GetFileSize() : Win32::Win32FSWrapper, IO::ZipFileEntry
GetFileWriteTime() : Win32::Win32FSWrapper, IO::IoServer
GetFloat() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlReader, Util::CmdLineArgs, Util::Variant, Models::Model, Models::ModelNode,
BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- GetFloat4() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlReader, Util::CmdLineArgs, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- GetFloat4Array() : Util::Variant
- GetFloat4AtIndex() : Util::CmdLineArgs
- GetFloat4DefValue() : Attr::AttrId
- GetFloatArray() : Util::Variant
- GetFloatAtIndex() : Util::CmdLineArgs
- GetFloatDefValue() : Attr::AttrId
- GetFocusEntity() : BaseGameFeature::FocusManager
- GetFormat() : Base::StreamTextureSaverBase, CoreGraphics::VertexComponent
- GetFourCC() : Attr::Attribute, Attr::AttributeDefinitionBase, Attr::AttrId, Core::Rtti
- GetFov() : AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
- GetFragGroupIndex() : Models::SkinShapeNode
- GetFragmentArray() : Models::SkinShapeNode
- GetFrameCount() : Graphics::GraphicsServer
- GetFramePassByIndex() : Frame::FrameShader
- GetFramePassByName() : Frame::FrameShader
- GetFrameShader() : Graphics::View
- GetFrameShaderByIndex() : Frame::FrameServer
- GetFrameShaderName() : AsyncGraphics::ViewProxy
- GetFrameTime() : App::AsyncRenderApplication, App::RenderApplication
- GetFromURI() : Interface::CopyFile
- GetGlobalBoundingBox() : Graphics::GraphicsEntity
- GetGlobalLightShadowBufferTexture() : Lighting::SM30ShadowServer
- GetGraphicsEntities() : BaseGameFeature::EnvEntityManager
- GetGravity() : Models::ParticleSystemNode
- GetGuid() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, Util::Variant, CoreGraphics::AdapterInfo, Models::Model,
Models::ModelNode, BaseGameFeature::GlobalAttrsManager, Game::Entity
- GetGuidArray() : Util::Variant
- GetGuidDefValue() : Attr::AttrId
- GetHandlerAtIndex() : Messaging::Port
- GetHandlerByIndex() : IO::Console
- GetHeight() : Base::RenderTargetBase, Base::TextureBase, CoreGraphics::DisplayMode
- GetHelp() : Scripting::Command
- GetHighFrequencyVibrator() : Base::GamePadBase
- GetHomeDirectory() : Win32::Win32FSWrapper
- GetHorizontalRotation() : GraphicsFeature::CameraOrbit
- GetHostAddr() : Win32::Win32IpAddress
- GetHostName() : Win32::Win32IpAddress
- GetHour() : Base::CalendarTimeBase
- GetHttpMethod() : Http::HttpRequestReader
- GetHwnd() : Win32::Win32DisplayDevice
- GetIconName() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- GetIndexBuffer() : Base::MeshBase, Base::RenderDeviceBase
- GetIndexBufferDepth() : CoreGraphics::CPUIndexBuffer
- GetIndexType() : Base::IndexBufferBase
- GetInput() : IO::Console, IO::ConsoleHandler, Win32::Win32ConsoleHandler
- GetInputFocusEntity() : BaseGameFeature::FocusManager
- GetInstanceDataset() : BaseGameFeature::CategoryManager::Category
- GetInstanceEntity() : BaseGameFeature::CategoryManager
- GetInstances() : Models::Model
- GetInstancesByAttr() : BaseGameFeature::CategoryManager
- GetInstancesByAttrs() : BaseGameFeature::CategoryManager
- GetInstanceTable() : BaseGameFeature::CategoryManager
- GetInstanceTableName() : BaseGameFeature::CategoryManager::Category
- GetInt() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlReader, Util::_CmdLineArgs, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile,
BaseGameFeature::GlobalAttrsManager, Game::Entity
- GetIntArray() : Util::Variant
- GetIntAtIndex() : Util::CmdLineArgs
- GetIntDefValue() : Attr::AttrId
- GetInvLightProjTransform() : Lighting::AbstractLightEntity
- GetInvModelTransform() : Base::TransformDeviceBase
- GetInvModelViewTransform() : Base::TransformDeviceBase
- GetInvProjTransform() : Base::TransformDeviceBase
- GetInvTransform() : Lighting::AbstractLightEntity
- GetInvViewTransform() : Base::TransformDeviceBase
- GetJoint() : Graphics::ActorEntity, Models::CharacterNode
- GetJointIndex() : Models::SkinShapeNode
- GetJointIndexByName() : Graphics::ActorEntity
- GetJointMatrixByIndex() : Graphics::ActorEntity
- GetJointMatrixByName() : Graphics::ActorEntity
- GetJointPaletteSize() : Models::SkinShapeNode
- GetKeepAlive() : Win32::Win32Socket
- GetKey() : CoreGraphics::DisplayEvent, Input::InputEvent
- GetKeyboardCaptureHandler() : Base::InputServerBase
- GetLastError() : Win32::SysFunc
- GetLevelName() : App::GameStateHandler
- GetLightDir() : Lighting::PSSMUtil
- GetLightDirection() : Lighting::GlobalLightEntity
- GetLightingMode() : Frame::FrameBatch
- GetLightType() : Lighting::AbstractLightEntity
- GetLinks() : Graphics::GraphicsEntity
- GetLoader() : Resources::Resource
- GetLocalBoundingBox() : Graphics::GraphicsEntity
- GetLocalJointMatrix() : Graphics::ActorEntity
- GetLocalLightShadowBufferTexture() : Lighting::SM30ShadowServer
- GetLocalTime() : Base::CalendarTimeBase, Win32::Win32CalendarTime
- GetLocalTransform() : Models::TransformNodeInstance
- GetLoop() : Models::ParticleSystemNode
- GetLowFrequencyVibrator() : Base::GamePadBase
- GetMainRenderTarget() : Frame::FrameShader
- GetManagedAnimation() : Models::CharacterNode
- GetManagedMesh() : Models::ShapeNode
- GetMapperByResourceType() : Resources::ResourceManager
- GetMaterialUnderMouse() : BaseGameFeature::EnvQueryManager
- getmatrix() : Math::transform44
- GetMatrix44() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlReader, Util::CommandLineArgs, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::GlobalAttrsManager, Game::Entity
- GetMatrix44Array() : Util::Variant
- GetMatrix44AtIndex() : Util::CommandLineArgs
- GetMatrix44DefValue() : Attr::AttrId
- GetMaxMsgSize() : Win32::Win32Socket
- GetMaxNumPlayers() : Base::GamePadBase
- GetMaxProgressValue() : BaseGameFeature::LoaderServer
- GetMaxScreenSpaceSize() : Resources::ManagedResource
- GetMaxTriggerDistance() : BaseGameFeature::EntityManager
- GetMediaType() : IO::Stream
- GetMesh() : Resources::ManagedMesh
- GetMethod() : Http::HttpRequest
- GetMilliSecond() : Base::CalendarTimeBase
- GetMinute() : Base::CalendarTimeBase
- GetMipLevel() : Base::StreamTextureSaverBase
- GetModel() : Models::ManagedModel, Models::ModelInstance, Models::ModelNode
- GetModelEntity() : Models::ModelInstance
- GetModelInstance() : Graphics::ModelEntity, Models::ModelNodeInstance
- GetModelNode() : Models::ModelNodeInstance
- GetModelResId() : Models::ModelReader
- GetModelResourceMapper() : Models::ModelServer
- GetModelResourceState() : Graphics::ModelEntity
- GetModelTransform() : Base::TransformDeviceBase
- GetModelViewProjTransform() : Base::TransformDeviceBase
- GetModelViewTransform() : Base::TransformDeviceBase
- GetModifiedRowsExcludeNewAndDeletedRows() : Attr::AttributeTable
- GetModifiedTracking() : Attr::AttributeTable
- GetMonth() : Base::CalendarTimeBase
- GetMountedZipArchives() : IO::ZipFileSystem
- GetMouseButton() : CoreGraphics::DisplayEvent, Input::InputEvent
- GetMouseCaptureHandler() : Base::InputServerBase
- GetMouseExcludeSet() : BaseGameFeature::EnvQueryManager
- GetMouseMovement() : Win32::Win32InputServer
- GetMousePos3d() : BaseGameFeature::EnvQueryManager
- GetMovement() : Base::MouseBase, Win32::Win32Mouse
- GetMyThreadName() : Win32::Win32Thread
- GetName() : Frame::FramePass, Win32::Win32Thread, IO::Assign, Attr::Attribute, Attr::AttributeDefinitionBase, Attr::AttrId, Core::Rtti, Http::HttpRequestHandler, Win32::Win32Heap, Messaging::AsyncPort, Scripting::Command, AsyncGraphics::ViewProxy, Base::ShaderVariableBase, Base::ShaderVariationBase, Frame::FramePostEffect, Frame::FrameShader, Graphics::Stage, Graphics::View, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::CategoryManager::Category
- GetNearHeight() : AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
- GetNearWidth() : AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
- GetNewColumnIndices() : Attr::AttributeTable
- GetNewRowIndices() : Attr::AttributeTable
- GetNodeFilter() : Frame::FrameBatch
- GetNodeInstances() : Models::ModelInstance
- GetNoDelay() : Win32::Win32Socket
- GetNodes() : Models::Model
- GetNormMousePos() : CoreGraphics::DisplayEvent, Input::InputEvent
- GetNumArgs() : Scripting::ArgsBlock, Util::CmdLineArgs
- GetNumArrayElements() : Base::ShaderVariableBase
- GetNumBatches() : Frame::FramePass
- GetNumberOfIndices() : Resources::DynamicMeshResourceLoader
- GetNumberOfVertices() : Resources::DynamicMeshResourceLoader
- GetNumCategories() : `BaseGameFeature::CategoryManager`
- GetNumClips() : `Models::CharacterNode`
- GetNumColorBuffers() : `Base::RenderTargetBase`
- GetNumColumns() : `Attr::AttributeTable`
- GetNumCommands() : `Scripting::ScriptServer`
- GetNumComponents() : `Base::VertexLayoutBase`
- GetNumEntitiesInHierarchy() : `Graphics::Cell`
- GetNumEntitiesInHierarchyByType() : `Graphics::Cell`
- GetNumEntitiesInHierarchyByTypeMask() : `Graphics::Cell`
- GetNumFragments() : `Models::SkinShapeNode`
- GetNumFramePasses() : `Frame::FrameShader`
- GetNumHandlers() : `IO::Console, Messaging::Port`
- GetNumIndices() : `Base::IndexBufferBase, CoreGraphics::PrimitiveGroup`
- GetNumInstances() : `BaseGameFeature::CategoryManager`
- GetNumJoints() : `Models::CharacterNode`
- GetNumMipLevels() : `Base::TextureBase`
- GetNumPasses() : `Base::ShaderVariationBase`
- GetNumPostEffects() : `Frame::FrameShader`
- GetNumPrimitiveGroups() : `Base::MeshBase`
- GetNumPrimitives() : `CoreGraphics::PrimitiveGroup`
- GetNumRenderTargets() : `Frame::FrameShader`
- GetNumRows() : `Attr::AttributeTable`
- GetNumSharedVariables() : `Base::ShaderServerBase, Direct3D9::D3D9ShaderServer`
- GetNumSkins() : `Models::CharacterNode`
- GetNumStates() : `App::GameApplication`
- GetNumVariables() : `Base::ShaderInstanceBase, Frame::FrameBatch, Frame::FramePostEffect, Frame::FramePass`
- GetNumVariations() : `Base::ShaderInstanceBase, Models::CharacterNode`
- GetNumVertices() : `CoreGraphics::PrimitiveGroup, Base::VertexBufferBase`
- GetObject() : `Util::Proxy< TYPE >, Util::Variant`
- GetObjectRefCount() : `Util::Proxy< TYPE >`
- GetObtainFocus() : `GraphicsFeature::CameraFocus, GraphicsFeature::InputFocus`
- GetOptBool() : `IO::XmlReader`
- GetOptFloat() : IO::XmlReader
- GetOptFloat4() : IO::XmlReader
- GetOptInt() : IO::XmlReader
- GetOptMatrix44() : IO::XmlReader
- GetOptString() : IO::XmlReader
- GetOriginalShader() : Base::ShaderInstanceBase
- GetOverlayAnimation() : Graphics::ActorEntity
- GetOverlayAnimationDuration() : Graphics::ActorEntity
- GetOverlayClip() : Graphics::ActorEntity
- GetParent() : Core::Rtti, Models::ModelNode, Models::ModelNodeInstance
- GetParentCell() : Graphics::Cell
- GetParticleMesh() : Models::ParticleSystemNode
- GetPassword() : IO::ZipArchive
- GetPath() : IO::Assign
- GetPhysicsEntity() : PhysicsFeature::PhysicsProperty
- GetPixelFormat() : Base::TextureBase, CoreGraphics::DisplayMode
- GetPixelPosition() : Base::MouseBase
- GetPlaceholderResourceId() : Resources::ResourceMapper
- GetPlayerIndex() : Base::GamePadBase
- GetPort() : Win32::Win32IpAddress, Http::HttpServer
- getposition() : Math::transform44
- GetPosition() : IO::MemoryStream, IO::Stream, IO::FileStream, Models::TransformNode, Models::TransformNodeInstance, IO::ZipFileStream
- GetPostEffectByIndex() : Frame::FrameShader
- GetPostEffectByName() : Frame::FrameShader
- GetPreShaders() : Base::ShaderInstanceBase
- GetPrimitiveGroup() : Base::RenderDeviceBase
- GetPrimitiveGroupAtIndex() : Base::MeshBase
- GetPrimitiveTopology() : CoreGraphics::PrimitiveGroup
- GetPriority() : Resources::ManagedResource, Win32::Win32Thread
- GetProfileDirectory() : BaseGameFeature::UserProfile
- GetProfileRootDirectory() : BaseGameFeature::UserProfile
- GetProgressResource() : BaseGameFeature::LoaderServer
- GetProgressText() : BaseGameFeature::LoaderServer
- GetProjMapUvOffsetAndScale() : Lighting::AbstractLightEntity
- GetProjTransform(): Lighting::AbstractLightEntity, Base::TransformDeviceBase, Graphics::CameraEntity, AsyncGraphics::CameraEntityProxy
- GetPSSMSplitDistances(): Lighting::SM30ShadowServer
- GetPSSMSplitLightProjTransforms(): Lighting::SM30ShadowServer
- GetPtr(): Util::Blob
- GetReadWriteColumnIndices(): Attr::AttributeTable
- GetRecvBufSize(): Win32::Win32Socket
- GetRecvStream(): Net::TcpClient, Net::TcpClientConnection
- GetRefCount(): Core::RefCounted
- GetRelativeDistanceChange(): GraphicsFeature::CameraDistance
- GetRenderCount(): Resources::ManagedResource
- GetRenderOldestFirst(): Models::ParticleSystemNode
- GetRenderTarget(): Graphics::View, Frame::FramePass, Frame::FramePostEffect
- GetRenderTargetByIndex(): Frame::FrameShader
- GetRenderTargetByName(): Frame::FrameShader
- GetRequestURI(): Http::HttpRequestReader
- GetResolveRect(): Base::RenderTargetBase
- GetResolveTexture(): Base::RenderTargetBase
- GetResolveTextureHeight(): Base::RenderTargetBase
- GetResolveTextureResourceId(): Base::RenderTargetBase
- GetResource(): Resources::ResourceLoader, Resources::ResourceSaver, Resources::ManagedResource
- GetResourceId(): Graphics::ModelEntity, Resources::Resource, Resources::ManagedResource
- GetResourceState(): Models::ShapeNode, Models::ModelNode, Models::CharacterNode, Models::Model, Models::StateNode
- GetResourceType(): Resources::ManagedResource, Resources::SimpleResourceMapper, Resources::ResourceMapper
- GetResponseContentStream(): Http::HttpRequest
- GetResult(): Interface::CopyFile, Util::Crc, Interface::IOMessage
- GetResults(): Scripting::Command
- GetReturnCode() : App::Application
- GetReUseAddr() : Win32::Win32Socket
- GetRevision() : CoreGraphics::AdapterInfo
- GetRGBCurve() : Models::ParticleSystemNode
- GetRootCell() : Graphics::Stage
- GetRootLocation() : Http::HttpRequestHandler
- getrotate() : Math::transform44
- GetRotate() : Models::TransformNodeInstance, Models::TransformNode
- GetRotatePivot() : Models::TransformNode
- getrotatepivot() : Math::transform44
- GetRotatePivot() : Models::TransformNodeInstance
- GetRowUserData() : Attr::AttributeTable
- GetSaveGame() : App::GameStateHandler
- GetSaveGameDirectory() : BaseGameFeature::UserProfile
- GetSaveGamePath() : BaseGameFeature::UserProfile
- GetSaver() : Resources::Resource
- getscale() : Math::transform44
- GetScale() : Models::TransformNode, Models::TransformNodeInstance
- getscalepivot() : Math::transform44
- GetScalePivot() : Models::TransformNode, Models::TransformNodeInstance
- GetScreenPosition() : Base::MouseBase
- GetSecond() : Base::CalendarTimeBase
- GetSemantic() : Base::ShaderVariableBase
- GetSemanticIndex() : CoreGraphics::VertexComponent
- GetSemanticName() : CoreGraphics::VertexComponent
- GetSendBufSize() : Win32::Win32Socket
- GetSendStream() : Net::TcpClientConnection, Net::TcpClient
- GetServerAddress() : Net::TcpClient
- GetSetupMode() : App::GameStateHandler
- GetShader() : Frame::FrameBatch, Frame::FramePass, Frame::FramePostEffect
- GetShaderFeatures() : Frame::FrameBatch
- GetShaderInstance() : Models::StateNode
- GetShaderParamBindMode() : Base::ShaderServerBase
- GetShaderVariable() : Base::ShaderVariableInstanceBase
- GetShaderVariableInstance() : Models::StateNodeInstance
- GetShadowBufferUvOffsetAndScale(): 
  **Lighting::AbstractLightEntity**
- GetSharedResources(): **Resources::SharedResourceServer**
- GetSharedResourcesByType():
  **Resources::SharedResourceServer**
- GetSharedVariableByIndex(): **Base::ShaderServerBase**, **Direct3D9::D3D9ShaderServer**
- GetSharedVariableByName(): **Base::ShaderServerBase**, **Direct3D9::D3D9ShaderServer**
- GetSharedVariableBySemantic(): **Direct3D9::D3D9ShaderServer**, **Base::ShaderServerBase**
- GetSignature(): **CoreGraphics::VertexComponent**
- GetSize(): **IO::ZipFileStream**, **IO::Stream**, **IO::MemoryStream**
- GetSkinIndexByName(): **Models::CharacterNode**
- GetSkinInfoArray(): **Models::CharacterNode**
- GetSkinInfoAt(): **Models::CharacterNode**
- GetSkinList(): **Graphics::ActorEntity**
- GetSockAddr(): **Win32::Win32IpAddress**
- GetSortingMode(): **Frame::FrameBatch**
- GetSplitDistances(): **Lighting::PSSMUtil**
- GetSplitLightProjTransform(): **Lighting::PSSMUtil**
- GetSplitLightProjTransforms(): **Lighting::PSSMUtil**
- GetSplitLightTransform(): **Lighting::PSSMUtil**
- GetSplitProjTransform(): **Lighting::PSSMUtil**
- GetStackSize(): **Win32::Win32Thread**
- GetStage(): **Graphics::View**, **Graphics::GraphicsEntity**, **Graphics::Cell**
- GetStageBuilder(): **Graphics::Stage**
- GetStageByName(): **Graphics::GraphicsServer**
- GetStageName(): **AsyncGraphics::ViewProxy**
- GetStageProxies(): **AsyncGraphics::GraphicsServerProxy**
- GetStageProxy(): **AsyncGraphics::GraphicsEntityProxy**
- GetStageProxyByName(): **AsyncGraphics::GraphicsServerProxy**
- GetStages(): **Graphics::GraphicsServer**
- GetState(): **Resources::ManagedResource**, **Resources::Resource**, **Resources::ResourceLoader**
- GetStateHandlerAt(): **App::GameApplication**
- GetStatus() : **Http::HttpRequest**
- GetStream() : **IO::StreamWriter**, **Interface::WriteStream**, **Interface::ReadStream**, **IO::StreamReader**, **Base::StreamTextureSaverBase**
- GetStreamByteOrder() : **IO::BinaryWriter**, **IO::BinaryReader**
- GetStreamClassByUriScheme() : **IO::IoServer**
- GetString() : **Util::CmdLineArgs**, **Attr::AttributeContainer**, **BaseGameFeature::UserProfile**, **Util::Variant**, **Models::ModelNode**, **BaseGameFeature::GlobalAttrsManager**, **IO::XmlReader**, **Models::Model**, **Game::Entity**, **Attr::AttributeTable**, **Attr::AttributeContainer**, **Attr::Attribute**
- GetStringArray() : **Util::Variant**
- GetStringAtIndex() : **Util::CmdLineArgs**
- GetStringDefValue() : **Attr::AttrId**
- GetSubSystemId() : **CoreGraphics::AdapterInfo**
- GetSubType() : **IO::MediaType**
- GetSyntax() : **Scripting::Command**
- GetSystemTime() : **Win32::Win32CalendarTime**, **Base::CalendarTimeBase**
- GetTail() : **IO::URI**
- GetTargetEntityId() : **BaseGameFeature::MoveFollow**
- GetTempDirectory() : **Win32::Win32FSWrapper**
- GetTemplateDataset() : **BaseGameFeature::CategoryManager::Category**
- GetTemplateTable() : **BaseGameFeature::CategoryManager**
- GetTemplateTableName() : **BaseGameFeature::CategoryManager::Category**
- GetTexture() : **Resources::ManagedTexture**
- GetThreadPriority() : **Messaging::AsyncPort**
- GetThreadStackSize() : **Messaging::AsyncPort**
- GetTicks() : **Win32::Win32Timer**
- GetTime() : **Models::ModelInstance**, **Graphics::GraphicsEntity**, **App::AsyncRenderApplication**, **Win32::Win32Timer**, **App::RenderApplication**
- GetToURI() : **Interface::CopyFile**
- GetTransform() : **Graphics::GraphicsEntity**, **Models::ModelInstance**, **AsyncGraphics::GraphicsEntityProxy**
- `GetType()` : Frame::FrameBatch, Base::TextureBase, Util::Variant, AsyncGraphics::GraphicsEntityProxy, Graphics::GraphicsEntity, Models::ModelNode, Input::InputEvent, Base::ShaderVariableBase, IO::MediaType
- `GetUniqueId()` : Game::Entity
- `GetUpVector()` : BaseGameFeature::EnvQueryManager
- `GetURI()` : IO::ZipArchive, IO::Stream, Interface::IOMessage, Http::HttpRequest
- `GetUsage()` : Base::ResourceBase
- `GetUseCount()` : Resources::Resource
- `GetUserDirectory()` : Win32::Win32FSWrapper
- `GetUserProfile()` : BaseGameFeature::LoaderServer
- `GetValue()` : Attr::Attribute
- `GetValueType()` : Attr::AttributeDefinitionBase, Attr::AttrId, Attr::Attribute
- `GetVariableByIndex()` : Frame::FrameBatch, Frame::FramePass, Base::ShaderInstanceBase, Frame::FramePostEffect
- `GetVariableByName()` : Base::ShaderInstanceBase
- `GetVariableBySemantic()` : Base::ShaderInstanceBase
- `GetVariationByFeatureMask()` : Base::ShaderInstanceBase
- `GetVariationIndexByName()` : Models::CharacterNode
- `GetVariationJointsAt()` : Models::CharacterNode
- `GetVariationNameAt()` : Models::CharacterNode
- `GetVariationsUri()` : Models::CharacterNode
- `GetVendorId()` : CoreGraphics::AdapterInfo
- `GetVertexBuffer()` : Base::MeshBase, Base::RenderDeviceBase
- `GetVertexByteSize()` : Base::VertexLayoutBase
- `GetVertexLayout()` : Base::VertexBufferBase
- `GetVerticalRotation()` : GraphicsFeature::CameraOrbit
- `GetViewByName()` : Graphics::GraphicsServer
- `GetViewClass()` : AsyncGraphics::ViewProxy
- `GetViewName()` : AsyncGraphics::CameraEntityProxy
- `GetViewProjTransform()` : AsyncGraphics::CameraEntityProxy, Base::TransformDeviceBase, Graphics::CameraEntity
- `GetViewProxies()` : AsyncGraphics::GraphicsServerProxy
- **GetViewProxyByName()**:
  `AsyncGraphics::GraphicsServerProxy`
- **GetViews()**:
  `Graphics::GraphicsServer`
- **GetViewTransform()**:
  `Base::TransformDeviceBase`,
  `Graphics::CameraEntity`,
  `AsyncGraphics::CameraEntityProxy`
- **GetVisibleModelNodeInstances()**:
  `Models::ModelNode`,
  `Models::VisResolver`
- **GetVisibleModelNodes()**:
  `Models::VisResolver`
- **GetWaitForMessages()**:
  `Messaging::AsyncPort`
- **GetWeekday()**:
  `Base::CalendarTimeBase`
- **GetWidth()**:
  `CoreGraphics::DisplayMode`,
  `Base::RenderTargetBase`, `Base::TextureBase`
- **GetWindowTitle()**:
  `Base::DisplayDeviceBase`
- **GetXPos()**:
  `CoreGraphics::DisplayMode`
- **GetYear()**:
  `Base::CalendarTimeBase`
- **GetYPos()**:
  `CoreGraphics::DisplayMode`
- **GetZFar()**:
  `AsyncGraphics::CameraEntityProxy`,
  `Graphics::CameraEntity`
- **GetZNear()**:
  `Graphics::CameraEntity`,
  `AsyncGraphics::CameraEntityProxy`
- **GlobalAttrsManager()**:
  `BaseGameFeature::GlobalAttrsManager`
- **GlobalLightEntity()**:
  `Lighting::GlobalLightEntity`
- **GraphicsEntity()**:
  `Graphics::GraphicsEntity`
- **GraphicsEntityProxy()**:
  `AsyncGraphics::GraphicsEntityProxy`
- **GraphicsServer()**:
  `Graphics::GraphicsServer`
- **GraphicsServerProxy()**:
  `AsyncGraphics::GraphicsServerProxy`
- **gt()**:
  `Math::float2`
- **GuidAttrId()**:
  `Attr::GuidAttrId`
- h -

- Handled() : **Messaging::Message**
- HandleDelayedJobs() : **BaseGameFeature::EntityManager**
- HandleEvent() : **CoreGraphics::RenderEventHandler**, **CoreGraphics::ThreadSafeDisplayEventHandler**, **CoreGraphics::DisplayEventHandler**, **CoreGraphics::ThreadSafeRenderEventHandler**, **Win32::Win32InputDisplayEventHandler**
- HandleLeftMouseBtnDown() : **PhysicsFeature::MouseGripperProperty**
- HandleLeftMouseBtnUp() : **PhysicsFeature::MouseGripperProperty**
- HandleMessage() : **Messaging::Dispatcher**, **Messaging::Handler**, **Messaging::Port**, **AsyncGraphics::AsyncGraphicsHandler**, **Game::Property**, **PhysicsFeature::PhysicsProperty**
- HandlePendingEvents() : **CoreGraphics::ThreadSafeDisplayEventHandler**, **CoreGraphics::ThreadSafeRenderEventHandler**
- Handler() : **Messaging::Handler**
- HandleRequest() : **Debug::CorePageHandler**, **Http::DefaultHttpRequestHandler**, **Http::HttpRequestHandler**, **Debug::IoPageHandler**, **Debug::MemoryPageHandler**, **Debug::ScriptingPageHandler**, **Debug::DisplayPageHandler**, **Debug::MeshPageHandler**, **Debug::ShaderPageHandler**, **Debug::TexturePageHandler**
- HasActiveEntities() : **BaseGameFeature::EntityManager**
- HasArg() : **Scripting::ArgsBlock**, **Util::CmdLineArgs**
- HasAssign(): `IO::IoServer`
- HasAttr(): `Attr::AttributeContainer`, `IO::XmlReader`, `Models::Model`, `Models::ModelNode`, `BaseGameFeature::UserProfile`, `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`
- HasCategory(): `BaseGameFeature::CategoryManager`
- HasCharacter3Set(): `Graphics::ActorEntity`
- HasColorBuffer(): `Base::RenderTargetBase`
- HasColumn(): `Attr::AttributeTable`
- HasCommand(): `Scripting::ScriptServer`
- HasComponent(): `Base::VertexLayoutBase`
- HasContent(): `IO::XmlReader`
- HasDeletedRows(): `Attr::AttributeTable`
- HasDepthStencilBuffer(): `Base::RenderTargetBase`
- HasEntity(): `Game::Property`
- HasFramePass(): `Frame::FrameShader`
- HasFrameShader(): `Frame::FrameServer`
- HashCode(): `Util::Blob`, `Util::String`, `Win32::Win32Guid`
- HashTable(): `Util::HashTable<KEYTYPE, VALUETYPE>`
- HasIndexBuffer(): `Base::MeshBase`
- HasInput(): `IO::Console`, `IO::ConsoleHandler`, `Win32::Win32ConsoleHandler`
- HasInstanceDataset():
  - `BaseGameFeature::CategoryManager::Category`
- HasInstanceTable(): `BaseGameFeature::CategoryManager`
- HasManagedModel(): `Models::ModelServer`
- HasManagedResource(): `Resources::ResourceManager`
- HasMapper(): `Resources::ResourceManager`
- HasModifiedRows(): `Attr::AttributeTable`
- HasMouseIntersection():
  - `BaseGameFeature::EnvQueryManager`
- HasNewRows(): `Attr::AttributeTable`
- HasNode(): `IO::XmlReader`, `Models::Model`
- HasNodeInstance(): `Models::ModelInstance`
- HasParent(): `Util::SimpleTree<VALUETYPE>::Node`, `Models::ModelNode`, `Models::ModelNodeInstance`
- HasPort(): `Messaging::Dispatcher`
- HasPostEffect(): `Frame::FrameShader`
- HasRecvData(): `Win32::Win32Socket`
- HasRenderTarget(): `Frame::FrameShader`
- HasResolveTexture(): `Base::RenderTargetBase`
- HasShader(): `Base::ShaderServerBase`
- HasShaderVariableInstance(): `Models::StateNodeInstance`
- HasSharedResource(): `Resources::SharedResourceServer`
- HasSharedVariableByName(): `Base::ShaderServerBase`, `Direct3D9::D3D9ShaderServer`
- HasSharedVariableBySemantic(): `Base::ShaderServerBase`, `Direct3D9::D3D9ShaderServer`
- HasStage(): `Graphics::GraphicsServer`
- HasStageProxy(): `AsyncGraphics::GraphicsServerProxy`
- HasStarted(): `Game::GameServer`
- HasStream(): `IO::StreamReader`, `IO::StreamWriter`
- HasTemplateDataset():
  - `BaseGameFeature::CategoryManager::Category`
- HasTemplateTable(): `BaseGameFeature::CategoryManager`
- HasVariableByName(): `Base::ShaderInstanceBase`
- HasVariableBySemantic(): `Base::ShaderInstanceBase`
- HasVariation(): `Base::ShaderInstanceBase`
- HasVertexBuffer(): `Base::MeshBase`
- HasView(): `Graphics::GraphicsServer`
- HasViewProxy(): `AsyncGraphics::GraphicsServerProxy`
- height(): `Math::rectangle<TYPE>`
- Height(): `Util::FixedTable<TYPE>`
- HorizontalRule(): `Http::HtmlPageWriter`
- Host(): `IO::URI`
- HtmlPageWriter(): `Http::HtmlPageWriter`
- HttpRequest(): `Http::HttpRequest`
- HttpRequestHandler(): `Http::HttpRequestHandler`
- HttpRequestReader(): `Http::HttpRequestReader`
- HttpServer(): `Http::HttpServer`
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
- i -

- Id() : Messaging::Id
- IncrClientCount() : Resources::ManagedResource
- Increment() : Win32::Win32Interlocked
- IncrUseCount() : Resources::Resource
- IndexBufferBase() : Base::IndexBufferBase
- Initialize() : Util::QuadTree< TYPE >::Node
- InputEvent() : Input::InputEvent
- InputFocus() : GraphicsFeature::InputFocus
- InputHandler() : Input::InputHandler
- InputServer() : Input::InputServer
- InputServerBase() : Base::InputServerBase
- Insert() : Threading::SafePriorityQueue< PRITYPE, TYPE >, Util::Array< TYPE >, Util::SimpleTree< VALUETYPE >::Node
- InsertEntity() : Graphics::Cell
- InsertSorted() : Util::Array< TYPE >
- inside() : Math::rectangle< TYPE >, Math::sphere
- Instance() : Core::Factory
- IntAttrId() : Attr::IntAttrId
- intersect() : Math::line
- intersect_sweep() : Math::sphere
- intersects() : Math::bbox, Math::sphere
- IMessage() : Interface::IOMessage
- IoPageHandler() : Debug::IoPageHandler
- IoServer() : IO::IoServer
- IsA() : Core::RefCounted
- IsActive() : Graphics::GraphicsEntity, Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property
- IsAlwaysOnTop() : `AsyncGraphics::DisplayProxy`, `Base::DisplayDeviceBase`
- IsArray() : `Base::ShaderVariableBase`
- IsAsyncEnabled() : `Resources::Resource`, `Resources::ResourceMapper`
- IsAttached() : `Input::InputHandler`
- IsAttachedToCell() : `Graphics::GraphicsEntity`
- IsAttachedToModel() : `Models::ModelInstance`, `Models::ModelNode`
- IsAttachedToModelInstance() : `Models::ModelNodeInstance`
- IsAttachedToResource() : `Resources::ResourceLoader`, `Resources::ResourceSaver`
- IsAttachedToResourceManager() : `Resources::ResourceMapper`
- IsAttachedToServer() : `Graphics::Stage`, `Graphics::View`
- IsAttachedToStage() : `Graphics::Cell`, `Graphics::GraphicsEntity`
- IsAttachedToStageProxy() : `AsyncGraphics::GraphicsEntityProxy`
- IsAttachedToView() : `Graphics::CameraEntity`
- IsBlocking() : `Net::TcpClient`
- IsBound() : `Win32::Win32Socket`
- IsCapturing() : `Input::InputHandler`
- IsConnected() : `Win32::Win32Socket`, `Base::GamePadBase`, `Net::TcpClient`, `Net::TcpClientConnection`
- IsDefaultRenderTarget() : `Base::RenderTargetBase`
- IsDefaultView() : `AsyncGraphics::ViewProxy`
- Is Derived From() : `Core::Rtti`
- IsDeviceName() : `Win32::Win32FSWrapper`
- isdirty() : `Math::transform44`
- IsDisplayModeSwitchEnabled() : `AsyncGraphics::DisplayProxy`, `Base::DisplayDeviceBase`
- IsDynamic() : `Attr::AttributeDefinitionBase`, `Attr::AttrId`
- IsEmpty() : `IO::URI`, `Scripting::ArgsBlock`, `Threading::SafePriorityQueue<PRITYPE, TYPE>`, `Threading::SafeQueue<TYPE>`, `Util::Array<TYPE>`, `Util::Dictionary<KEYTYPE, VALUETYPE>`, `Util::HashTable<KEYTYPE, VALUETYPE>`, `Util::List<TYPE>`, `Util::Queue<TYPE>`, `Util::SimpleTree<VALUETYPE>::Node`, `Util::Stack<..`
TYPE > , Util::String
  IsEnabled() : PhysicsFeature::PhysicsProperty
  IsEntityHandleValid() : AsyncGraphics::GraphicsEntityProxy
  IsFullscreen() : AsyncGraphics::DisplayProxy , Base::DisplayDeviceBase
  IsInBeginFrame() : Base::RenderDeviceBase
  IsInetAddr() : Win32::Win32IpAddress
  IsInFocus() : BaseGameFeature::EntityManager
  IsInstanceOf() : Core::RefCounted
  IsLoaded() : Resources::Resource , BaseGameFeature::UserProfile
  IsMapped() : IO::Stream
  IsMemoryMappingEnabled() : IO::BinaryWriter , IO::BinaryReader
  IsModified() : Attr::AttributeTable
  IsMounted() : IO::ZipFileSystem
  isOpen() : Frame::FrameServer , Graphics::GraphicsServer ,
             Base::InputServerBase , Models::ModelServer ,
             Models::VisResolver , Resources::ResourceManager ,
             Base::TransformDeviceBase , Lighting::LightServerBase ,
             BaseGameFeature::LoaderServer ,
             CoreGraphics::VertexLayoutServer ,
             Base::RenderDeviceBase , Scripting::ScriptServer ,
             IO::Stream , Core::CoreServer , App::Application ,
             Http::HttpServer , IO::Console , IO::ConsoleHandler ,
             IO::StreamReader , IO::StreamWriter , IO::ZipArchive ,
             Messaging::AsyncPort , Messaging::Handler , Net::TcpServer ,
             Win32::Win32Socket , Resources::SharedResourceServer ,
             Base::ShapeRendererBase , AsyncGraphics::DisplayProxy ,
             AsyncGraphics::GraphicsServerProxy ,
             Base::DisplayDeviceBase , Base::ShaderServerBase
  IsOrthogonal() : Graphics::CameraEntity ,
                   AsyncGraphics::CameraEntityProxy
  IsOverlayAnimationActive() : Graphics::ActorEntity
  IsPending() : Resources::Resource
  IsPerspective() : Graphics::CameraEntity ,
                   AsyncGraphics::CameraEntityProxy
  IsQuitRequested() : App::RenderApplication ,
                     Base::InputServerBase , Game::GameServer ,
App::AsyncRenderApplication
- IsReadOnly(): Win32::Win32FSWrapper, IO::IoServer
- IsRegistered(): Scripting::Command
- IsResolved(): Models::VisResolveContainer< TYPE >
- IsRowDeleted(): Attr::AttributeTable
- IsRowModified(): Attr::AttributeTable
- IsRunning(): Win32::Win32Thread
- IsSpecial(): BaseGameFeature::CategoryManager::Category
- IsTripleBufferingEnabled(): Base::DisplayDeviceBase, AsyncGraphics::DisplayProxy
- IsUriSchemeRegistered(): IO::IoServer
- IsValid(): Util::Atom< TYPE >, IO::MediaType, IO::URI, Attr::AttrId, Graphics::GraphicsEntity, Util::Proxy< TYPE >, BaseGameFeature::CategoryManager::Entry, AsyncGraphics::GraphicsEntityProxy, AsyncGraphics::ViewProxy, Util::String, Win32::Win32Guid, Base::RenderTargetBase, Util::FourCC, Base::VertexLayoutBase, Util::Blob, Base::ShaderInstanceBase
- IsValidBool(): Util::String
- IsValidFloat(): Util::String
- IsValidFloat4(): Util::String
- IsValidFourCC(): Attr::AttrId
- IsValidHttpRequest(): Http::HttpRequestReader
- IsValidInt(): Util::String
- IsValidMatrix44(): Util::String
- IsValidName(): Attr::AttrId
- IsVerticalSyncEnabled(): AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- IsVirtual(): BaseGameFeature::CategoryManager::Category
- IsZipArchiveMounted(): IO::IoServer
- IsZipFileSystemEnabled(): IO::IoServer
- 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
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
- k -

- Key() : Util::KeyValuePair< KEYTYPE, VALUETYPE >
- KeyAtIndex() : Util::Dictionary< KEYTYPE, VALUETYPE >
- KeyboardBase() : Base::KeyboardBase
- KeyDown() : Base::KeyboardBase
- KeyPressed() : Base::KeyboardBase
- KeysAsArray() : Util::Dictionary< KEYTYPE, VALUETYPE >
- KeyUp() : Base::KeyboardBase
- KeyValuePair() : Util::KeyValuePair< KEYTYPE, VALUETYPE >
- KeyValuePairAtIndex() : Util::Dictionary< KEYTYPE, VALUETYPE >

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
- I -

- u
- v
- w
- x
- y
- z
- ~

- i -

- le() : Math::float2
- Leave() : Win32::Win32CriticalSection
- length() : Math::line
- Length() : Util::String
- length() : Math::float2
- lengthsq() : Math::float2, Math::line
- Level() : Util::QuadTree< TYPE >::Node
- LevelExists() : BaseGameFeature::CategoryManager
- LightServer() : Lighting::LightServer
- LightServerBase() : Lighting::LightServerBase
- line() : Math::line
- LineBreak() : Http::HtmlPageWriter
- List() : Util::List< TYPE >
- ListDirectories() : IO::IoServer, Win32::Win32FSWrapper, IO::ZipArchive
- Listen() : Win32::Win32Socket
- ListFiles() : IO::IoServer, Win32::Win32FSWrapper, IO::ZipArchive
- Load() : Resources::Resource,
  BaseGameFeature::EntityLoader,
  BaseGameFeature::EntityLoaderBase,
  BaseGameFeature::EnvironmentLoader,
  BaseGameFeature::LevelLoader,
  BaseGameFeature::UserProfile
- LoadAttributes() : BaseGameFeature::GlobalAttrsManager
- LoadEntities() : BaseGameFeature::LoaderServer
- LoaderServer() : BaseGameFeature::LoaderServer
- **LoadFailed()**: `Resources::Resource`
- **LoadFrameShaders()**: `Frame::FrameShaderLoader`
- **LoadFromAttrs()**: `Models::Model`, `Models::ModelNode`, `Models::TransformNode`
- **LoadInstances()**: `BaseGameFeature::CategoryManager`
- **LoadLevel()**: `BaseGameFeature::LoaderServer`
- **LoadManagedModel()**: `Models::ModelServer`
- **LoadResources()**: `Models::ParticleSystemNode`, `Models::SkinShapeNode`, `Models::ModelNode`, `Models::Model`, `Models::CharacterNode`, `Models::StateNode`, `Models::ShapeNode`
- **LoadVariation()**: `Models::CharacterNode`
- **LoadXmlTable()**: `Attr::AttributeTable`
- **LocalPath()**: `IO::URI`
- **LocalTimeToFileTime()**: `Base::CalendarTimeBase`, `Win32::Win32CalendarTime`
- **LookupManagedModel()**: `Models::ModelServer`
- **LookupManagedResource()**: `Resources::ResourceManager`
- **LookupNode()**: `Models::Model`
- **LookupNodeInstance()**: `Models::ModellInstance`
- **LookupNodeInstanceByRTTI()**: `Models::ModellInstance`
- **LookupSharedResource()**: `Resources::SharedResourceServer`
- **lt()**: `Math::float2`
- **LuaServer()**: `Scripting::LuaServer`
- m -

- ManagedResource() : Resources::ManagedResource
- Manager() : Game::Manager
- Map() : IO::MemoryStream, Base::TextureBase, Base::VertexBufferBase, IO::Stream, CoreGraphics::CPUIndexBuffer, CoreGraphics::CPUVertexBuffer, IO::FileStream, IO::ZipFileStream, Direct3D9::D3D9IndexBuffer, Direct3D9::D3D9Texture, Base::IndexBufferBase, Direct3D9::D3D9VertexBuffer
- MapCubeFace() : Direct3D9::D3D9Texture, Base::TextureBase
- MapInfo() : Base::TextureBase::MapInfo
- MatchPattern() : Util::String
- Matrix44AttrId() : Attr::Matrix44AttrId
- maximize() : Math::float2
- MayaCameraUtil() : RenderUtil::MayaCameraUtil
- MediaType() : IO::MediaType
- MemoryIndexBufferLoaderBase() :
  Base::MemoryIndexBufferLoaderBase
- MemoryPageHandler() : Debug::MemoryPageHandler
- MemoryStream() : IO::MemoryStream
- MemoryVertexBufferLoaderBase() :
  Base::MemoryVertexBufferLoaderBase
- MeshBase() : Base::MeshBase
- MeshPageHandler() : Debug::MeshPageHandler
- Message() : Messaging::Message
- MessageReader() : Messaging::MessageReader
- MessageWriter() : Messaging::MessageWriter
- minimize() : Math::float2
- Model() : Models::Model
- ModelEntity() : Graphics::ModelEntity
- ModelInstance() : Models::ModelInstance
- ModelNode() : Models::ModelNode
- ModelNodeInstance() : Models::ModelNodeInstance
- ModelServer() : Models::ModelServer
- MonthToString() : Base::CalendarTimeBase
- Mount() : IO::ZipFileSystem
- MountZipArchive() : IO::IoServer
- MouseBase() : Base::MouseBase
- MouseGripperProperty() : PhysicsFeature::MouseGripperProperty
- MoveFollow() : BaseGameFeature::MoveFollow
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
- n -

- N2ModelReader() : Models::N2ModelReader
- Node() : Util::QuadTree< TYPE >::Node , Util::SimpleTree< VALUETYPE >::Node
- normalize() : Math::float2
- NotifyEventHandlers() : Base::RenderDeviceBase , Base::DisplayDeviceBase
- NotifyGameLoad() : Game::GameServer
- NotifyGameSave() : Game::GameServer
- NotifyVisible() : Models::ModelInstance
- nullvec() : Math::vector
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
ObtainKeyboardCapture() : Base::InputServerBase
ObtainMouseCapture() : Base::InputServerBase
OnActivate() : Graphics::GraphicsEntity,
    BaseGameFeature::GlobalAttrsManager, Game::Entity,
    Graphics::ModelEntity, Game::FeatureUnit, Game::Manager,
    Graphics::ActorEntity,
    BaseGameFeature::CategoryManager, Game::Property,
    PhysicsFeature::MouseGripperProperty,
    BaseGameFeature::EnvQueryManager,
    PhysicsFeature::PhysicsProperty
OnApply() : PreShaders::GaussianBlur5x5FilterKernel,
    PreShaders::BoxFilterKernel
OnAttach() : CoreGraphics::DisplayEventHandler,
    CoreGraphics::RenderEventHandler, Base::GamePadBase,
    Base::KeyboardBase, Base::MouseBase,
    Input::InputHandler, PreShaders::BoxFilterKernel,
    PreShaders::GaussianBlur5x5FilterKernel
OnAttachToCell() : Graphics::GraphicsEntity
OnAttachToModel() : Models::ModellInstance,
    Models::ModelNode
OnAttachToModellInstance() : Models::StateNodeInstance,
    Models::TransformNodeInstance,
    Models::ModelNodeInstance,
    Models::CharacterNodeInstance,
    Models::ParticleSystemNodeInstance,
    Models::SkinShapeNodeInstance
OnAttachToResource() :
Resources::DynamicMeshResourceLoader, Resources::ResourceLoader, Resources::ResourceSaver
- OnAttachToResourceManager() : Resources::ResourceMapper, Resources::SimpleResourceMapper
- OnAttachToServer() : Graphics::Stage, Graphics::View
- OnAttachToStage() : Graphics::Cell, Graphics::GraphicsEntity
- OnAttachToView() : Graphics::CameraEntity
- OnBeginFrame() : Base::KeyboardBase, Base::MouseBase, Input::InputHandler, XInput::XInputGamePad, BaseGameFeature::EntityManager, Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property, PhysicsFeature::MouseGripperProperty
- OnChar() : Win32::Win32DisplayDevice
- OnCloseRequested() : Win32::Win32DisplayDevice
- OnConfigureDisplay() : App::AsyncRenderApplication
- OnConfigureDisplayDevice() : App::RenderApplication
- OnCreateHandlers() : AsyncHttp::AsyncHttpInterface, Messaging::AsyncPort, AsyncGraphics::AsyncGraphicsInterface
- OnCreateManagedResource() : Resources::ResourceMapper, Resources::SimpleResourceMapper
- OnDeactivate() : Graphics::CameraEntity, Graphics::GraphicsEntity, Graphics::ModelEntity, BaseGameFeature::CategoryManager, BaseGameFeature::EntityManager, BaseGameFeature::EnvEntityManager, BaseGameFeature::EnvQueryManager, Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property, PhysicsFeature::MouseGripperProperty, PhysicsFeature::PhysicsProperty, Graphics::ActorEntity
- OnDetach() : PreShaders::BoxFilterKernel, PreShaders::GaussianBlur5x5FilterKernel
- OnDiscardManagedResource() : Resources::ResourceMapper, Resources::SimpleResourceMapper
- OnEndFrame() : Input::InputHandler, BaseGameFeature::EntityManager, Game::FeatureUnit, Game::Manager
- OnEvent() : Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- OnExecute() : Scripting::Command
- OnFrame() : AsyncGraphics::GraphicsServerProxy, Graphics::GraphicsServer, Base::InputServerBase, Win32::Win32InputServer, App::GameStateHandler, BaseGameFeature::EnvQueryManager, BaseGameFeature::FocusManager, Game::FeatureUnit, Game::GameServer, Game::Manager
- OnGainActivity() : Game::Entity, Game::Property
- OnHide() : Graphics::GraphicsEntity
- OnKeyDown() : Win32::Win32DisplayDevice
- OnKeyUp() : Win32::Win32DisplayDevice
- OnKillFocus() : Win32::Win32DisplayDevice
- OnLoad() : Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property
- OnLoadCancelled() : Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, Models::StreamModelLoader, Resources::ResourceLoader
- OnLoadRequested() : CoreGraphics::CPUMemoryIndexBufferLoader, CoreGraphics::CPUMemoryVertexBufferLoader, Direct3D9::D3D9MemoryIndexBufferLoader, Direct3D9::D3D9MemoryVertexBufferLoader, Direct3D9::D3D9StreamShaderLoader, Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, Models::StreamModelLoader, Resources::DynamicMeshResourceLoader, Resources::ResourceLoader
- OnLoseActivity() : Game::Entity, Game::Property
- OnLostDevice() : Direct3D9::D3D9ShaderInstance
- OnMinimized() : Win32::Win32DisplayDevice
- OnMouseButton() : Win32::Win32DisplayDevice
- OnMouseMove() : Win32::Win32DisplayDevice
- OnMouseWheel() : Win32::Win32DisplayDevice
- OnMoveAfter() : Game::Entity, Game::Property, PhysicsFeature::MouseGripperProperty, PhysicsFeature::PhysicsProperty
- OnMoveBefore() : Game::Entity, Game::Property, PhysicsFeature::MouseGripperProperty
- OnNotifyVisible() : Models::ModelNodeInstance, Models::ParticleSystemNodeInstance, Models::ShapeNodeInstance, Models::SkinShapeNodeInstance
- OnObtainCapture() : Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- OnPaint() : Win32::Win32DisplayDevice
- OnPending() : Direct3D9::D3D9StreamTextureLoader, CoreGraphics::StreamAnimationLoader, CoreGraphics::StreamMeshLoader, Models::StreamModelLoader, Resources::ResourceLoader
- OnPrepare() : Resources::ResourceMapper, Resources::SimpleResourceMapper
- OnProcessInput() : App::AsyncRenderApplication, App::AsyncViewerApplication, App::RenderApplication, App::ViewerApplication
- OnRegister() : Scripting::Command
- OnReleaseCapture() : Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- OnRemove() : CoreGraphics::DisplayEventHandler, CoreGraphics::RenderEventHandler, Input::InputHandler
- OnRemoveFromCell() : Graphics::GraphicsEntity
- OnRemoveFromModel() : Models::ModelInstance, Models::ModelNode
- OnRemoveFromModelInstance() : Models::ModelNodeInstance, Models::CharacterNodeInstance, Models::ParticleSystemNodeInstance, Models::SkinShapeNodeInstance, Models::StateNodeInstance, Models::TransformNodeInstance
- OnRemoveFromResource() : Resources::DynamicMeshResourceLoader, Resources::ResourceLoader, Resources::ResourceSaver
- OnRemoveFromResourceManager() : Resources::ResourceMapper, Resources::SimpleResourceMapper
- OnRemoveFromServer() : Graphics::Stage, Graphics::View
- **OnRemoveFromStage()**: Graphics::Cell, Graphics::GraphicsEntity
- **OnRemoveFromView()**: Graphics::CameraEntity
- **OnRender()**: Graphics::GraphicsEntity, Game::Entity, Game::Property
- **OnRenderDebug()**: Graphics::GraphicsEntity, Graphics::ModelEntity, BaseGameFeature::EntityManager, Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property, PhysicsFeature::MouseGripperProperty
- **OnRenderFrame()**: App::RenderApplication, App::ViewerApplication
- **OnReset()**: Base::GamePadBase, Base::KeyboardBase, Base::MouseBase, Input::InputHandler
- **OnResetDevice()**: Direct3D9::D3D9ShaderInstance
- **OnRestored()**: Win32::Win32DisplayDevice
- **OnSave()**: Base::StreamTextureSaverBase, Direct3D9::D3D9StreamTextureSaver, Resources::ResourceSaver, BaseGameFeature::EntityManager, Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property
- **OnSetCursor()**: Win32::Win32DisplayDevice
- **OnSetFocus()**: Win32::Win32DisplayDevice
- **OnShow()**: Graphics::GraphicsEntity
- **OnStart()**: BaseGameFeature::EntityManager, Game::Entity, Game::FeatureUnit, Game::Manager, Game::Property
- **OnStateEnter()**: App::GameStateHandler
- **OnStateLeave()**: App::GameStateHandler
- **OnToggleFullscreenWindowed()**: Win32::Win32DisplayDevice
- **OnUnregister()**: Scripting::Command
- **OnUpdateFrame()**: App::AsyncRenderApplication,
App::AsyncViewerApplication, App::RenderApplication

- Open() : IO::ZipFileStream, IO::ZipArchive,
  Resources::SharedResourceServer, Messaging::AsyncPort,
  Messaging::Handler, Net::TcpServer, Models::ModelWriter,
  Win32::Win32Socket, Scripting::LuaServer,
  Scripting::ScriptServer, App::AsyncRenderApplication,
  App::AsyncViewerApplication, App::RenderApplication,
  App::ViewerApplication, Lighting::SM30LightServer,
  AsyncGraphics::AsyncGraphicsHandler,
  AsyncGraphics::AsyncGraphicsInterface,
  AsyncGraphics::DisplayProxy, Graphics::GraphicsServer,
  AsyncGraphics::GraphicsServerProxy,
  Base::DisplayDeviceBase, Base::RenderDeviceBase,
  Base::ShaderServerBase, Base::ShapeRendererBase,
  Base::TransformDeviceBase, Direct3D9::D3D9RenderDevice,
  Direct3D9::D3D9ShaderServer,
  Direct3D9::D3D9ShapeRenderer,
  CoreGraphics::VertexLayoutServer,
  Win32::Win32DisplayDevice, Frame::FrameServer,
  Base::InputServerBase, Win32::Win32InputServer,
  Lighting::LightServerBase, Lighting::SM30ShadowServer,
  Models::BinaryModelReader, Models::ModelReader,
  Models::N2ModelReader, Models::XmlModelReader,
  Models::ModelServer, Models::BinaryModelWriter,
  Models::XmlModelWriter, Models::VisResolver,
  Resources::ResourceManager, App::GameApplication,
  BaseGameFeature::LoaderServer, IO::XmlWriter,
  Game::GameServer, App::Application,
  App::ConsoleApplication, IO::Stream,
  AsyncHttp::AsyncHttpInterface, Core::CoreServer,
  Http::HtmlPageWriter, Http::HttpServer, IO::BinaryReader,
  IO::BinaryWriter, IO::Console, IO::ConsoleHandler,
  IO::FileStream, IO::MemoryStream, IO::StreamReader,
  IO::StreamWriter, IO::XmlReader

- OpenDInputMouse() : Win32::Win32InputServer
- OpenFile() : Win32::Win32FSWrapper
- OpenProgressIndicator() : BaseGameFeature::LoaderServer
- OpenWindow() : Win32::Win32DisplayDevice
- operator *() : Math::vector, Math::float2, Util::List< TYPE
::Iterator
- operator *(): Math::vector, Math::float2
- operator bool(): Util::List< TYPE >::Iterator
- operator delete(): Util::Blob, Util::String, Win32::Win32Guid
- operator new(): Util::Blob, Util::String, Win32::Win32Guid
- operator !=(): Attr::Attribute, Attr::AttrId, Attr::BoolAttrId, Attr::Float4AttrId, Attr::FloatAttrId, Attr::IntAttrId, Attr::Matrix44AttrId, Attr::StringAttrId, IO::MediaType, IO::URI, Math::point, Math::vector, Math::float2, Util::Array< TYPE >, Util::Atom< TYPE >, Util::Blob, Util::FixedArray< TYPE >, Util::FixedTable< TYPE >, Util::KeyValuePair< KEYTYPE, VALUETYPE >, Util::List< TYPE >::Iterator, Util::Proxy< TYPE >, Util::Queue< TYPE >, Util::Stack< TYPE >, Util::Variant, CoreGraphics::DisplayMode, Win32::Win32Guid, Util::Variant, Core::Rtti, Util::FourCC, Util::Atom< TYPE >, Attr::Attribute, Attr::GuidAttrId, Attr::BlobAttrId
- operator +(): Math::float2, Math::point, Math::vector
- operator ++(): Util::List< TYPE >::Iterator
- operator +=(): Math::vector, Util::String, Math::point, Math::float2
- operator -(): Math::vector, Math::float2, Math::point, Math::float2
- operator --(): Util::List< TYPE >::Iterator
- operator -(): Math::point, Math::vector, Math::float2
- operator ->(): Util::List< TYPE >::Iterator
- operator <(): Util::Proxy< TYPE >, Attr::AttrId, Util::Blob, Win32::Win32IpAddress, Util::Atom< TYPE >, Util::KeyValuePair< KEYTYPE, VALUETYPE >, Util::Proxy< TYPE >, Util::Atom< TYPE >, Util::FourCC, Win32::Win32Guid
- operator <=(): Util::Proxy< TYPE >, Util::Blob, Util::Atom< TYPE >, Util::KeyValuePair< KEYTYPE, VALUETYPE >, Util::Proxy< TYPE >, Attr::AttrId, Util::FourCC, Win32::Win32Guid
- operator =(): Attr::Attribute, Util::Variant, Attr::Attribute, Util::Variant, Attr::Attribute, Util::Variant, Util::KeyValuePair< KEYTYPE, VALUETYPE >, Util::Atom< TYPE >, Util::Variant, Threading::SafePriorityQueue<
operator==( ) : Util::Variant , Util::Atom< TYPE > , Attr::Float4AttrId , Attr::Attribute , Attr::GuidAttrId , CoreGraphics::DisplayMode , Util::Proxy< TYPE > , Attr::BoolAttrId , Attr::StringAttrId , Attr::FloatAttrId , Math::float2 , Attr::Attribute , Util::FixedTable< TYPE > , Util::FixedArray< TYPE > , Math::point , Util::KeyValuePair< KEYTYPE , VALUETYPE > , Util::Atom< TYPE > , Util::List< TYPE >::Iterator , Util::Queue< TYPE > , Util::Variant , Util::FixedTable< TYPE > , Util::Variant , IO::MediaType , Util::Variant , Util::FixedArray< TYPE > , Util::Variant , Util::Blob , IO::URI , Threading::SafeQueue< TYPE > , Util::Array< TYPE > , Math::polar
operator>() : Win32::Win32Guid , Util::Variant , Util::Proxy< TYPE > , Win32::Win32IpAddress , Util::FourCC , IO::MediaType
operator>( ) : Win32::Win32IpAddress , Util::Proxy< TYPE > , Win32::Win32Guid , Util::Blob , Attr::AttrId , Attr::Attribute , Messaging::Id , Util::Variant , Util::Stack< TYPE > , Util::Variant , Attr::Attribute , Util::Variant , Util::Proxy< TYPE > , Util::Variant , Attr::Attribute , Util::Variant , Attr::Attribute , Util::List< TYPE >::Iterator , Attr::Attribute , Util::Atom< TYPE > , Core::Rtti , Util::Queue< TYPE > , Util::Variant , IO::URI , Attr::BlobAttrId , Win32::Win32IpAddress , Util::FourCC , IO::MediaType
operator>=( ) : Util::Proxy< TYPE > , Util::Blob , Util::KeyValuePair< KEYTYPE , VALUETYPE > , Util::Atom< TYPE > , Attr::AttrId , Util::Atom< TYPE > , Util::FourCC , Util::Proxy< TYPE > , Win32::Win32Guid
operator[]( ) : Util::HashTable< KEYTYPE , VALUETYPE > , Util::Stack< TYPE > , Util::SimpleTree< VALUETYPE >::Node , Util::Dictionary< KEYTYPE , VALUETYPE > , Util::SimpleTree<
VALUETYPE >::Node, Util::Dictionary< KEYTYPE,
VALUETYPE >, Util::FixedArray< TYPE >, Util::Queue< TYPE 
>, Util::Array< TYPE >, Util::String

- origin() : Math::point
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
- p -

- Parent() : Util::SimpleTree< VALUETYPE >::Node
- ParseBluePrints() : BaseGameFeature::FactoryManager
- ParseQuery() : IO::URI
- ParticleSystemNode() : Models::ParticleSystemNode
- ParticleSystemNodeInstance() :
  Models::ParticleSystemNodeInstance
- Peek() : Threading::SafePriorityQueue< PRITYPE, TYPE >,
  Threading::SafeQueue< TYPE >,
  Win32::Win32Event,
  Util::Queue< TYPE >,
  Util::Stack< TYPE >,
  Messaging::AsyncPort
- PerformGarbageCollection() : Util::Atom< TYPE >
- PhysicsProperty() : PhysicsFeature::PhysicsProperty
- point() : Math::point
- pointat() : Math::line
- polar() : Math::polar
- Pop() : Util::Stack< TYPE >
- Port() : IO::URI
- Prepare() : Resources::ResourceManager
- Present() : Base::RenderDeviceBase,
  Direct3D9::D3D9RenderDevice
- PrimitiveGroup() : CoreGraphics::PrimitiveGroup
- Print() : IO::Console, IO::ConsoleHandler,
  Win32::Win32ConsoleHandler, IO::Console
- PrintCommandHelp() : Scripting::ScriptServer
- PrintCommandList() : Scripting::ScriptServer
- PrintDebug() : Attr::AttributeTable
- ProcessWindowMessages() : Base::DisplayDeviceBase,
Win32::Win32DisplayDevice
- project_screen_rh() : Math::sphere
- Property() : Game::Property
- Proxy() : Util::Proxy< TYPE >
- PSSMUtil() : Lighting::PSSMUtil
- Push() : Util::Stack< TYPE >
- PutEvent() : CoreGraphics::RenderEventHandler, CoreGraphics::ThreadSafeRenderEventHandler, CoreGraphics::DisplayEventHandler, Base::InputServerBase, CoreGraphics::ThreadSafeDisplayEventHandler

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
- q -

- QuadtreeStageBuilder()
- Query()
- Queue()

Graphics::QuadtreeStageBuilder
IO::URI
Util::Queue< TYPE >

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
Read() : IO::FileStream, IO::MemoryStream, Win32::Win32FSWrapper, IO::ZipFileStream, IO::Stream, IO::ZipFileEntry, System::Win32Registry
ReadAll() : IO::TextReader
ReadAllLines() : IO::TextReader
ReadBlob() : IO::BinaryReader
ReadBool() : IO::BinaryReader
ReadChar() : IO::BinaryReader, IO::TextReader
ReadDInputMouse() : Win32::Win32InputServer
ReadDouble() : IO::BinaryReader
ReadFloat() : IO::BinaryReader
ReadFloat4() : IO::BinaryReader
ReadGuid() : IO::BinaryReader
ReadInt() : IO::BinaryReader
ReadLine() : IO::TextReader
ReadMatrix44() : IO::BinaryReader
ReadMessage() : Messaging::MessageReader
ReadRequest() : Http::HttpRequestReader
ReadShort() : IO::BinaryReader
ReadString() : IO::BinaryReader
ReadUChar() : IO::BinaryReader
ReadUInt() : IO::BinaryReader
ReadUShort() : IO::BinaryReader
Realloc() : Win32::Win32Heap
Reallocate() : Util::Array<Type>
rectangle() : Math::rectangle<Type>
Recv() : Net::TcpClient, Net::TcpClientConnection,
Win32::Win32Socket
- RecvFrom() : Win32::Win32Socket
- RefCounted() : Core::RefCounted
- Register() : Attr::AttributeDefinitionBase, Core::Factory
- RegisterCommand() : Scripting::LuaServer, Scripting::ScriptServer
- RegisterDynamicAttribute() : Attr::AttributeDefinitionBase
- RegisterMessage() : Messaging::Port
- RegisterPropertyCallback() : Game::Entity
- RegisterSharedResource() : Resources::SharedResourceServer
- RegisterUriScheme() : IO::IoServer
- Release() : Core::RefCounted, Util::SimpleTree< VALUETYPE >::Node
- ReleaseKeyboardCapture() : Base::InputServerBase
- ReleaseMouseCapture() : Base::InputServerBase
- Remove() : Util::List< TYPE >
- RemoveAllLoaders() : BaseGameFeature::LoaderServer
- RemoveAllMappers() : Resources::ResourceManager
- RemoveBack() : Util::List< TYPE >
- RemoveDisplayEventHandler() : AsyncGraphics::DisplayProxy
- RemoveEntity() : Graphics::Cell, Graphics::Stage, BaseGameFeature::EntityManager
- RemoveEntityFromTriggered() : BaseGameFeature::EntityManager
- RemoveEntityImmediate() : BaseGameFeature::EntityManager
- RemoveEntityLoader() : BaseGameFeature::LoaderServer
- RemoveEventHandler() : Base::DisplayDeviceBase, Base::RenderDeviceBase
- RemoveFront() : Util::List< TYPE >
- RemoveGameFeature() : Game::GameServer
- RemoveHandler() : IO::Console, Messaging::Port
- RemoveInputHandler() : Base::InputServerBase
- RemoveManager() : Game::FeatureUnit
- RemoveMapper() : Resources::ResourceManager
- RemoveNode() : Models::Model
- RemoveNullEntriesFromArrays() : BaseGameFeature::EntityManager
- RemovePort() : Messaging::Dispatcher
- RemovePreShader() : Base::ShaderInstanceBase
- RemoveRenderEventHandler() : AsyncGraphics::DisplayProxy
- RemoveRequestHandler() : Http::HttpServer
- RenameLevel() : BaseGameFeature::CategoryManager
- Render() : Frame::FrameBatch, Frame::FramePostEffect, Frame::FrameShader, Graphics::View, Models::ModelNodeInstance, Models::ParticleSystemNodeInstance, Models::ShapeNodeInstance, Models::SkinShapeNodeInstance, Frame::FramePass
- RenderApplication() : App::RenderApplication
- RenderDebug() : Graphics::View, Models::ModelInstance, Models::ModelNodeInstance, Game::GameServer, Models::CharacterNodeInstance
- RenderDebugSimple() : Graphics::View
- RenderDevice() : CoreGraphics::RenderDevice
- RenderDeviceBase() : Base::RenderDeviceBase
- RenderEvent() : CoreGraphics::RenderEvent
- RenderEventHandler() : CoreGraphics::RenderEventHandler
- RenderFragment() : Models::SkinShapeNodeInstance
- RenderSkinning() : Models::SkinShapeNodeInstance
- RenderTargetBase() : Base::RenderTargetBase
- ReplaceChars() : Util::String
- ReplaceIllegalFilenameChars() : Util::String
- RequestLoadResources() : Models::SkinShapeNode
- RequestQuit() : Game::GameServer
- RequestState() : App::GameApplication
- RequestUnloadResources() : Models::SkinShapeNode
- Reserve() : Util::Blob, Util::Array< TYPE >, Util::String
- ReserveRows() : Attr::AttributeTable
- Reset() : Models::VisResolveContainer< TYPE >, Util::Array< TYPE >, Win32::Win32Timer, Base::InputServerBase, RenderUtil::MayaCameraUtil
- ResetFeatureBits() : Base::ShaderServerBase
- ResetModifiedState() : Attr::AttributeTable
- ResolveAssigns() : IO::IoServer
- ResolveAssignsInString() : IO::IoServer
- ResolveVisibleLights() : Graphics::View
- ResolveVisibleModelNodeInstances() : Graphics::View
- Resource() : Resources::Resource
- ResourceBase() : Base::ResourceBase
- ResourceLoader() : Resources::ResourceLoader
- ResourceManager() : Resources::ResourceManager
- ResourceMapper() : Resources::ResourceMapper
- ResourceSaver() : Resources::ResourceSaver
- Root() : Util::SimpleTree< VALUE_TYPE >
- Row() : Util::QuadTree< TYPE >::Node
- RowIndex() : BaseGameFeature::CategoryManager::Entry
- Rtti() : Core::Rtti
- Run() : App::GameApplication, App::AsyncRenderApplication, App::RenderApplication, App::Application
- Running() : Win32::Win32Timer
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
SafePriorityQueue() : Threading::SafePriorityQueue<PRITYPE, TYPE>
SafeQueue() : Threading::SafeQueue<TYPE>
Save() : Resources::Resource, BaseGameFeature::UserProfile
SaveAttributes() : BaseGameFeature::GlobalAttrsManager
SaveModel() : Models::ModelServer
SaveScreenshot() : Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
SaveToAttrs() : Models::Model, Models::ModelNode, Models::TransformNode
Scheme() : IO::URI
ScriptingPageHandler() : Debug::ScriptingPageHandler
ScriptServer() : Scripting::ScriptServer
Seek() : IO::FileStream, IO::MemoryStream, IO::Stream, Win32::Win32FSWrapper, IO::ZipFileStream
SelectActiveVariation() : Base::ShaderInstanceBase, Direct3D9::D3D9ShaderInstance
SemanticNameToString() : CoreGraphics::VertexComponent
Send() : Messaging::AsyncPort, Messaging::Port, Net::TcpClient, Net::TcpClientConnection, Win32::Win32Socket
SendCreateMsg() : AsyncGraphics::GraphicsEntityProxy
SendMsg() : AsyncGraphics::GraphicsEntityProxy
SendSync() : Game::Entity
SendTo() : Win32::Win32Socket
SendWait() : Messaging::AsyncPort
- Set(): `IO::MediaType`, `IO::URI`, `Util::Array<TYPE>`, `Util::Blob`, `Util::FixedTable<TYPE>`, `Util::String`  
- set(): `Math::bbox`, `Math::point`, `Math::vector`, `Math::float2`, `Math::line`, `Math::polar`, `Math::rectangle<TYPE>`
- `Math::sphere`
- SetAbsMousePos(): `Input::InputEvent`  
- SetAccess(): `Base::ResourceBase`  
- SetAccessMode(): `IO::Stream`  
- SetAccessPattern(): `IO::Stream`  
- SetActiveShaderInstance(): `Base::ShaderServerBase`  
- SetActivityDistance(): `Models::ParticleSystemNode`  
- SetAdapter(): `AsyncGraphics::DisplayProxy`, `Base::DisplayDeviceBase`  
- SetAddress(): `Net::TcpServer`, `Win32::Win32Socket`  
- SetAllNodeInstancesVisible(): `Models::ModelInstance`  
- SetAlwaysOnTop(): `AsyncGraphics::DisplayProxy`, `Base::DisplayDeviceBase`  
- SetAnim(): `Models::CharacterNode`  
- SetAnimationMapping(): `Graphics::ActorEntity`  
- SetAntiAliasQuality(): `AsyncGraphics::DisplayProxy`, `Base::DisplayDeviceBase`, `Base::RenderTargetBase`  
- SetAppName(): `App::Application`, `Core::CoreServer`  
- SetAssign(): `IO::IoServer`  
- SetAsyncEnabled(): `Resources::Resource`, `Resources::ResourceManager`  
- SetAttr(): `Attr::AttributeContainer`, `Attr::AttributeTable`, `Models::Model`, `Models::ModelNode`, `Game::Entity`  
- SetAttributes(): `Graphics::StageBuilder`  
- SetAttrId(): `Attr::Attribute`  
- SetBackLightColor(): `Lighting::GlobalLightEntity`  
- SetBaseAnimation(): `Graphics::ActorEntity`  
- SetBaseIndex(): `CoreGraphics::PrimitiveGroup`  
- SetBaseVertex(): `CoreGraphics::PrimitiveGroup`  
- SetBillboardOrientation(): `Models::ParticleSystemNode`  
- SetBlob(): `Attr::Attribute`, `Attr::AttributeContainer`, `Attr::AttributeTable`, `Util::Variant`, `Models::Model`, `Models::ModelNode`, `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`  
- SetBlobArray(): `Util::Variant`
- `SetBlocking()`:
  - `Net::TcpClient`, `Win32::Win32Socket`
- `SetBool()`:
  - `Attr::Attribute`, `Attr::AttributeContainer`, `Attr::AttributeTable`, `IO::XmlWriter`, `Util::String`, `Util::Variant`, `Base::ShaderVariableBase`, `Base::ShaderVariableInstanceBase`, `Direct3D9::D3D9ShaderVariable`, `Models::Model`, `Models::ModelNode`, `BaseGameFeature::UserProfile`, `BaseGameFeature::GlobalAttrsManager`, `Game::Entity`
- `SetBoolArray()`:
  - `Util::Variant`, `Base::ShaderVariableBase`, `Base::ShaderVariableInstanceBase`, `Direct3D9::D3D9ShaderVariable`
- `SetBoundingBox()`:
  - `CoreGraphics::PrimitiveGroup`, `Graphics::Cell`, `Models::Model`, `Models::ModelNode`
- `SetBroadcast()`:
  - `Win32::Win32Socket`
- `SetCameraEntity()`:
  - `Graphics::View`, `Lighting::PSSMUtil`
- `SetCameraFocusEntity()`:
  - `BaseGameFeature::FocusManager`
- `SetCameraFocusToNextEntity()`:
  - `BaseGameFeature::FocusManager`
- `SetCastShadows()`:
  - `Lighting::AbstractLightEntity`
- `SetChar()`:
  - `Input::InputEvent`
- `SetCharPtr()`:
  - `Util::String`
- `SetChildrenVisibility()`:
  - `Models::ModelNodeInstance`
- `SetClearColor()`:
  - `Base::RenderTargetBase`, `Frame::FramePass`
- `SetClearDepth()`:
  - `Base::RenderTargetBase`, `Frame::FramePass`
- `SetClearStencil()`:
  - `Base::RenderTargetBase`, `Frame::FramePass`
- `SetClip()`:
  - `Models::CharacterNode`
- `SetCmdLineArgs()`:
  - `App::Application`, `Game::FeatureUnit`
- `SetColor()`:
  - `Lighting::AbstractLightEntity`
- `SetCompanyName()`:
  - `App::Application`, `Core::CoreServer`
- `SetContent()`:
  - `Http::HttpResponseWriter`
- `SetCoreId()`:
  - `Win32::Win32Thread`
- `SetCurve()`:
  - `Models::ParticleSystemNode`
- `SetDataPtr()`:
  - `Util::QuadTree<TYPE>::Node`
- `SetDay()`:
  - `Base::CalendarTimeBase`
- `SetDebugTextEnabled()`:
  - `BaseGameFeature::LoaderServer`
- `SetDefaultRenderTarget()`:
  - `Base::RenderTargetBase`
- SetDefaultView() : Graphics::GraphicsServer
- SetDepth() : Base::TextureBase
- SetDesc() : Http::HttpRequestHandler
- SetDescription() : CoreGraphics::AdapterInfo
- SetDeviceId() : CoreGraphics::AdapterInfo
- SetDeviceName() : CoreGraphics::AdapterInfo
- SetDisplayMode() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- SetDisplayModeSwitchEnabled() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- SetDistance() : BaseGameFeature::MoveFollow
- SetDontLinger() : Win32::Win32Socket
- SetDriverName() : CoreGraphics::AdapterInfo
- SetDriverVersionHighPart() : CoreGraphics::AdapterInfo
- SetDriverVersionLowPart() : CoreGraphics::AdapterInfo
- SetEmissionDuration() : Models::ParticleSystemNode
- SetEmitter() : Models::ParticleSystemNodeInstance
- SetEnabled() : PhysicsFeature::PhysicsProperty
- SetEntities() : GraphicsFeature::GetGraphicsEntities
- SetEntity() : Game::Property, GraphicsFeature::GetGraphicsEntities
- SetEnvEntityTransform() : BaseGameFeature::EnvEntityManager
- SetError() : Scripting::Command, Scripting::ScriptServer
- SetFadeAnimationMix() : Graphics::ActorEntity
- SetFeatureBits() : Base::ShaderServerBase
- SetFeatureMask() : Base::ShaderVariationBase
- SetFloat() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlWriter, Util::String, Util::Variant, Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- SetFloat4() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlWriter, Util::String, Util::Variant, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- SetFloat4Array() : Util::Variant
- SetFloatArray() : Util::Variant, Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable
- SetFocusEntity() : BaseGameFeature::FocusManager
- SetFocusToNextEntity() : BaseGameFeature::FocusManager
- SetFormat() : Base::StreamTextureSaverBase
- SetFragGroupIndex() : Models::SkinShapeNode
- SetFragment() : IO::URI
- SetFrameShader() : Graphics::View
- SetFromString() : Util::FourCC
- SetFromUInt() : Util::FourCC
- SetFromURI() : Interface::CopyFile
- SetFullscreen() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- SetGravity() : Models::ParticleSystemNode
- SetGuid() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, Util::Variant, CoreGraphics::AdapterInfo, Models::Model, Models::ModelNode, BaseGameFeature::GlobalAttrsManager, Game::Entity
- SetGuidArray() : Util::Variant
- SetHandled() : Messaging::Message
- SetHeight() : Base::RenderTargetBase, Base::TextureBase, CoreGraphics::DisplayMode
- SetHighFrequencyVibrator() : Base::GamePadBase
- SetHorizontalRotation() : GraphicsFeature::CameraOrbit
- SetHost() : IO::URI
- SetHostName() : Win32::Win32IpAddress
- SetHour() : Base::CalendarTimeBase
- SetIconName() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- SetIndexAccessMode() : Resources::DynamicMeshResourceLoader
- SetIndexBuffer() : Base::MeshBase, Base::RenderDeviceBase, Direct3D9::D3D9RenderDevice
- SetIndexBufferDepth() : CoreGraphics::CPUIndexBuffer
- SetIndexBufferType() : Resources::DynamicMeshResourceLoader
- SetIndexData() : Resources::DynamicMeshResourceLoader
- SetIndexType() : Base::IndexBufferBase
- SetIndexUsage() : Resources::DynamicMeshResourceLoader
- SetInputFocusEntity() : BaseGameFeature::FocusManager
- SetInputFocusToNextEntity() : BaseGameFeature::FocusManager
- SetInstanceEntity() : BaseGameFeature::CategoryManager
- SetInt() : Attr::Attribute, Attr::AttributeContainer, Attr::AttributeTable, IO::XmlWriter, Util::String, Util::Variant, Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable, Models::Model, Models::ModelNode, BaseGameFeature::UserProfile, BaseGameFeature::GlobalAttrsManager, Game::Entity
- SetIntArray() : Util::Variant, Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable
- SetInvisible() : Models::ParticleSystemNode
- SetJoint() : Models::CharacterNode
- SetJointIndex() : Models::SkinShapeNode
- SetJointIndices() : Models::SkinShapeNode
- SetKeepAlive() : Win32::Win32Socket
- SetKey() : Input::InputEvent
- SetLevelName() : App::GameStateHandler
- SetLightDir() : Lighting::PSSMUtil
- SetLightingMode() : Frame::FrameBatch
- SetLightType() : Lighting::AbstractLightEntity
- SetLoader() : Resources::Resource
- SetLocalBoundingBox() : Graphics::GraphicsEntity
- SetLocalPath() : IO::URI
- SetLoop() : Models::ParticleSystemNode
- SetLowFrequencyVibrator() : Base::GamePadBase
- SetMainRenderTarget() : Frame::FrameShader
- SetManagedResourceClass() : Resources::SimpleResourceMapper
- SetMatrix() : Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable
- SetMatrix44() : Attr::Attribute, Attr::AttributeContainer,
SetMatrix44Array() : Util::Variant
SetMatrixArray() : Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable
SetMaxProgressValue() : BaseGameFeature::LoaderServer
SetMaxTriggerDistance() : BaseGameFeature::EntityManager
SetMediaType() : IO::Stream
SetMemoryMappingEnabled() : IO::BinaryReader, IO::BinaryWriter
SetMethod() : Http::HttpRequest
SetMilliSecond() : Base::CalendarTimeBase
SetMinute() : Base::CalendarTimeBase
SetMipLevel() : Base::StreamTextureSaverBase
SetMipMapsEnabled() : Base::RenderTargetBase
SetModel() : Models::ModelReader
SetModelEntity() : Models::ModelInstance
SetModelResId() : Models::ModelReader
SetModelResourceMapper() : Models::ModelServer
SetModelTransform() : Base::TransformDeviceBase
SetModifiedTracking() : Attr::AttributeTable
SetMonth() : Base::CalendarTimeBase
SetMouseButton() : Input::InputEvent
SetMouseExcludeSet() : BaseGameFeature::EnvQueryManager
SetMouseMovement() : RenderUtil::MayaCameraUtil
SetName() : Http::HttpRequestHandler, Messaging::AsyncPort, Win32::Win32Thread, Base::ShaderVariableBase, Base::ShaderVariationBase, Frame::FramePass, Frame::FramePostEffect, Frame::FrameShader, Graphics::Stage, Graphics::View, Models::ModelNode, BaseGameFeature::UserProfile
SetNodeFilter() : Frame::FrameBatch
SetNoDelay() : Win32::Win32Socket
SetNormMousePos() : Input::InputEvent
SetNumArrayElements() : Base::ShaderVariableBase
SetNumberOfIndices() :
Resources::DynamicMeshResourceLoader
- SetNumberOfVertices() : Resources::DynamicMeshResourceLoader
- SetNumIndices() : Base::IndexBufferBase, CoreGraphics::PrimitiveGroup
- SetNumMipLevels() : Base::TextureBase
- SetNumPasses() : Base::ShaderVariationBase
- SetNumVertices() : Base::VertexBufferBase, CoreGraphics::PrimitiveGroup
- SetObject() : Util::Variant
- SetObtainFocus() : GraphicsFeature::CameraFocus, GraphicsFeature::InputFocus
- SetOrbitButton() : RenderUtil::MayaCameraUtil
- SetOrbiting() : RenderUtil::MayaCameraUtil
- SetOverlayAnimation() : Graphics::ActorEntity
- SetPanButton() : RenderUtil::MayaCameraUtil
- SetPanning() : RenderUtil::MayaCameraUtil
- SetParent() : Models::ModelNode, Models::ModelNodeInstance
- SetParticleRotationRandomize() : Models::ParticleSystemNode
- SetParticleSizeRandomize() : Models::ParticleSystemNode
- SetParticleStretch() : Models::ParticleSystemNode
- SetParticleVelocityRandomize() : Models::ParticleSystemNode
- SetPassword() : IO::ZipArchive
- SetPixelFormat() : Base::TextureBase, CoreGraphics::DisplayMode
- SetPlaceholder() : Resources::ManagedResource
- SetPlaceholderResourceId() : Resources::ResourceMapper
- SetPort() : IO::URI, Http::HttpServer, Win32::Win32IpAddress
- SetPosition() : Models::TransformNodeInstance
- setposition() : Math::transform44
- SetPosition() : Models::TransformNode
- SetPrecalcTime() : Models::ParticleSystemNode
- SetPrimitiveGroup() : Base::RenderDeviceBase
- SetPrimitiveGroups() : Base::MeshBase
- SetPrimitiveTopology() : CoreGraphics::PrimitiveGroup
- SetPriority() : Win32::Win32Thread, Resources::ManagedResource
- SetProgressResource() : BaseGameFeature::LoaderServer
SetProgressText() : BaseGameFeature::LoaderServer
SetProjMapUvOffsetAndScale() : Lighting::AbstractLightEntity
SetProjTransform() : Base::TransformDeviceBase
SetQuadTreeSettings() : Graphics::QuadtreeStageBuilder
SetQuery() : IO::URI
SetQuitRequested() : App::AsyncRenderApplication, App::RenderApplication, Base::InputServerBase
SetRandomRotDir() : Models::ParticleSystemNode
SetReadOnly() : IO::IoServer, Win32::Win32FSWrapper
SetRecvBufSize() : Win32::Win32Socket
SetRelativeDistanceChange() : GraphicsFeature::CameraDistance
SetRenderOldestFirst() : Models::ParticleSystemNode
SetRenderTarget() : Base::RenderTargetBase
SetResolveTextureHeight() : Base::RenderTargetBase
SetResolveTextureResourceId() : Base::RenderTargetBase
SetResolveTextureWidth() : Base::RenderTargetBase
SetResource() : Resources::ManagedResource
SetResourceClass() : Resources::SimpleResourceMapper
SetResourceId() : Graphics::ModelEntity, Resources::ManagedResource, Resources::Resource
SetResourceLoaderClass() : Resources::SimpleResourceMapper
SetResourceType() : Resources::ManagedResource
SetResponseContentStream() : Http::HttpRequest
SetResult() : Interface::CopyFile, Interface::IOMessage
SetReturnCode() : App::Application
SetReUseAddr() : Win32::Win32Socket
SetRevision() : CoreGraphics::AdapterInfo
SetRGBCurve() : Models::ParticleSystemNode
SetRootCell() : Graphics::Stage
SetRootLocation() : Http::HttpRequestHandler
SetRotate() : Models::TransformNodeInstance
setrotate() : Math::transform44
SetRotate() : Models::TransformNode
setrotatepivot() : Math::transform44
- SetRotatePivot() : Models::TransformNode, Models::TransformNodeInstance
- SetRowUserData() : Attr::AttributeTable
- SetSaveGame() : App::GameStateHandler
- SetSaver() : Resources::Resource
- setscale() : Math::transform44
- SetScale() : Models::TransformNode, Models::TransformNodeInstance
- SetScalePivot() : Models::TransformNodeInstance
- setscalepivot() : Math::transform44
- SetScalePivot() : Models::TransformNode
- SetScheme() : IO::URI
- SetSecond() : Base::CalendarTimeBase
- SetSemantic() : Base::ShaderVariableBase
- SetSendBufSize() : Win32::Win32Socket
- SetServerAddress() : Net::TcpClient
- SetSetupMode() : App::GameStateHandler
- SetShader() : Frame::FramePass, Frame::FrameBatch, Frame::FramePostEffect
- SetShaderFeatures() : Frame::FrameBatch
- SetShaderParamBindMode() : Base::ShaderServerBase
- SetShadowBufferUvOffsetAndScale() : Lighting::AbstractLightEntity
- SetSize() : Util::FixedTable< TYPE >, IO::MemoryStream, IO::Stream, Util::FixedArray< TYPE >
- SetSkinList() : Graphics::ActorEntity
- SetSockAddr() : Win32::Win32IpAddress
- SetSortingMode() : Frame::FrameBatch
- SetStackSize() : Win32::Win32Thread
- SetStage() : Graphics::View
- SetStageBuilder() : Graphics::Stage
- SetStartDate() : Models::ParticleSystemNode
- SetStartRotationMax() : Models::ParticleSystemNode
- SetStartRotationMin() : Models::ParticleSystemNode
- SetState() : Resources::Resource, Resources::ResourceLoader, App::GameApplication
- SetStatus() : Http::HttpRequest
- SetStatusCode() : Http::HttpResponseWriter
- SetStream() : Interface::ReadStream, IO::StreamReader,
ID::StreamWriter, Messaging::MessageReader, Messaging::MessageWriter, Base::StreamTextureSaverBase, Interface::WriteStream
- SetStreamByteOrder() : IO::BinaryWriter, IO::BinaryReader
- SetStretchDetail() : Models::ParticleSystemNode
- SetStretchToStart() : Models::ParticleSystemNode
- SetString() : Game::Entity, BaseGameFeature::UserProfile, Models::Model, IO::XmlWriter, Util::Variant, Attr::AttributeContainer, Attr::Attribute, Attr::AttributeTable, Models::ModelNode, BaseGameFeature::GlobalAttrsManager, Attr::AttributeTable
- SetStringArray() : Util::Variant
- SetSubSystemId() : CoreGraphics::AdapterInfo
- SetTargetEntityId() : BaseGameFeature::MoveFollow
- SetTexture() : Base::ShaderVariableBase, Base::ShaderVariableInstanceBase, Direct3D9::D3D9ShaderVariable
- SetThreadPriority() : Messaging::AsyncPort
- SetThreadStackSize() : Messaging::AsyncPort
- SetTileTexture() : Models::ParticleSystemNode
- SetTime() : Models::ModelInstance, Graphics::GraphicsEntity
- setTitle() : Http::HtmlPageWriter
- SetToDefault() : BaseGameFeature::UserProfile
- SetToFirstChild() : IO::XmlReader
- SetToNextChild() : IO::XmlReader
- SetToNode() : IO::XmlReader
- SetToParent() : IO::XmlReader
- SetToURI() : Interface::CopyFile
- SetTransform() : Models::ModelInstance, Graphics::GraphicsEntity, AsyncGraphics::GraphicsEntityProxy
- SetTripleBufferingEnabled() : AsyncGraphics::DisplayProxy, Base::DisplayDeviceBase
- SetType() : AsyncGraphics::GraphicsEntityProxy, Graphics::GraphicsEntity, Models::ModelNode, Util::Variant, Frame::FrameBatch, Input::InputEvent, Base::ShaderVariableBase, Base::TextureBase
- Setup() : Win32::Win32Heap, AsyncGraphics::CameraEntityProxy,
AsyncGraphics::GraphicsEntityProxy, Base::MemoryIndexBufferLoaderBase, Base::RenderTargetBase, Base::VertexLayoutBase, Direct3D9::D3D9ShaderInstance, Win32::SysFunc, Frame::FramePostEffect, Base::ShaderInstanceBase, RenderUtil::MayaCameraUtil, Win32::Win32Guid, Direct3D9::D3D9VertexLayout, Direct3D9::D3D9RenderTarget, Base::MemoryIndexBufferLoaderBase, Base::MemoryVertexBufferLoaderBase, Util::String, Util::Blob
- SetupAcceptedMessages() : Messaging::Port, PhysicsFeature::PhysicsProperty
- SetupCallbacks() : PhysicsFeature::PhysicsProperty, PhysicsFeature::MouseGripperProperty, Game::Property
- SetupDefaultAttributes() : PhysicsFeature::PhysicsProperty, Game::Property, PhysicsFeature::MouseGripperProperty
- SetupFromD3D9CubeTexture() : Direct3D9::D3D9Texture
- SetupFromD3D9Texture() : Direct3D9::D3D9Texture
- SetupFromD3D9VolumeTexture() : Direct3D9::D3D9Texture
- SetupGameFeatures() : App::GameApplication
- SetupManagedTextureVariable() : Models::StateNode
- SetupMultiSampleType() : Direct3D9::D3D9RenderTarget
- SetupOrthogonal() : AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
- SetupPerspectiveFov() : AsyncGraphics::CameraEntityProxy, Graphics::CameraEntity
- SetupSkinInfos() : Models::CharacterNode
- SetupStateHandlers() : App::GameApplication
- SetURI() : Http::HttpRequest, IO::Stream, IO::ZipArchive, Interface::IOMessage
- SetUsage() : Base::ResourceBase
- SetUserInfo() : IO::URI
- SetUserProfile() : BaseGameFeature::LoaderServer
- SetValid() : Graphics::GraphicsEntity
- SetValue() : Attr::Attribute
- SetValueFromString() : Attr::Attribute
- SetVariation() : Models::CharacterNode
- SetVariationsUri() : Models::CharacterNode
- SetVector(): `Base::ShaderVariableInstanceBase`, `Base::ShaderVariableBase`, `Direct3D9::D3D9ShaderVariable`
- SetVectorArray(): `Base::ShaderVariableInstanceBase`, `Direct3D9::D3D9ShaderVariable`, `Base::ShaderVariableBase`
- SetVendorId(): `CoreGraphics::AdapterInfo`
- SetVertexAccessMode(): `Resources::DynamicMeshResourceLoader`
- SetVertexBuffer(): `Base::MeshBase`, `Direct3D9::D3D9RenderDevice`, `Base::RenderDeviceBase`
- SetVertexComponents(): `Resources::DynamicMeshResourceLoader`
- SetVertexData(): `Resources::DynamicMeshResourceLoader`
- SetVertexLayout(): `Base::VertexBufferBase`
- SetVertexUsage(): `Resources::DynamicMeshResourceLoader`
- SetVertexWidth(): `Resources::DynamicMeshResourceLoader`
- SetVerticalRotation(): `GraphicsFeature::CameraOrbit`
- SetVerticalSyncEnabled(): `Base::DisplayDeviceBase`, `AsyncGraphics::DisplayProxy`
- SetViewAngleFade(): `Models::ParticleSystemNode`
- SetViewName(): `AsyncGraphics::CameraEntityProxy`
- SetViewTransform(): `Base::TransformDeviceBase`
- SetVisible(): `Graphics::GraphicsEntity`, `Models::ModelNodeInstance`, `Models::SkinShapeNodeInstance`, `AsyncGraphics::GraphicsEntityProxy`
- SetWaitForMessages(): `Messaging::AsyncPort`
- SetWeekday(): `Base::CalendarTimeBase`
- SetWidth(): `Base::TextureBase`, `CoreGraphics::DisplayMode`, `Base::RenderTargetBase`
- SetWindowTitle(): `Base::DisplayDeviceBase`
- SetXPos(): `CoreGraphics::DisplayMode`
- SetYear(): `Base::CalendarTimeBase`
- SetYPos(): `CoreGraphics::DisplayMode`
- SetZipFileSystemEnabled(): `IO::IoServer`
- SetZoomButton(): `RenderUtil::MayaCameraUtil`
- SetZoomIn(): `RenderUtil::MayaCameraUtil`
- SetZoomInButton(): `RenderUtil::MayaCameraUtil`
- SetZoomOut(): `RenderUtil::MayaCameraUtil`
- SetZoomOutButton(): `RenderUtil::MayaCameraUtil`
- ShaderBase() : Base::ShaderBase
- ShaderFeature() : CoreGraphics::ShaderFeature
- 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
- SharedResourceServer() : Resources::SharedResourceServer
- Shutdown() : Net::TcpClientConnection
- Signal() : Threading::SafeQueue< TYPE >, Win32::Win32Event, Threading::SafePriorityQueue< PRITYPE, TYPE >
- SimpleResourceMapper() : Resources::SimpleResourceMapper
- SimpleStageBuilder() : Graphics::SimpleStageBuilder
- SimpleTree() : Util::SimpleTree< VALUETYPE >
- Size() : Util::HashTable< KEYTYPE, VALUETYPE >, Threading::SafePriorityQueue< PRITYPE, TYPE >, Util::Stack< TYPE >, Threading::SafeQueue< TYPE >, Util::SimpleTree< VALUETYPE >::Node, Util::Queue< TYPE >, Util::FixedArray< TYPE >, Util::Blob, Util::Dictionary< KEYTYPE, VALUETYPE >
- size() : Math::bbox
- Size() : Util::Array< TYPE >, Util::List< TYPE >
- SizeOf() : CoreGraphics::IndexType
- SkinShapeNode() : Models::SkinShapeNode
- SkinShapeNodeInstance() : Models::SkinShapeNodeInstance
- Sleep() : Win32::SysFunc
- SM30LightServer() : Lighting::SM30LightServer
- SM30ShadowServer() : Lighting::SM30ShadowServer
- Sort() : Util::FixedArray< TYPE >, Util::Array< TYPE >
SortIfDirty() : Util::Dictionary< KEYTYPE, VALUETYPE >
sphere() : Math::sphere
SpotLightEntity() : Lighting::SpotLightEntity
Stack() : Util::Stack< TYPE >
Stage() : Graphics::Stage
StageBuilder() : Graphics::StageBuilder
Start() : Win32::Win32Timer
start() : Math::line
Start() : Game::GameServer, Win32::Win32Thread
StartEntities() : BaseGameFeature::EntityManager
StateNode() : Models::StateNode
StateTreeNodeInstance() : Models::StateTreeNodeInstance
Stop() : Win32::Win32Thread, Win32::Win32Timer, Game::GameServer
StopOverlayAnimation() : Graphics::ActorEntity
Stream() : IO::Stream
StreamAnimationLoader() : CoreGraphics::StreamAnimationLoader
StreamMeshLoader() : CoreGraphics::StreamMeshLoader
StreamModelLoader() : Models::StreamModelLoader
StreamReader() : IO::StreamReader
StreamTextureSaverBase() : Base::StreamTextureSaverBase
StreamWriter() : IO::StreamWriter
String() : Util::String
StringAttrId() : Attr::StringAttrId
StringToMonth() : Base::CalendarTimeBase
StringToType() : Util::Variant
StringToValueType() : Attr::Attribute
StringToWeekday() : Base::CalendarTimeBase
Strip() : Util::String
StripFileExtension() : Util::String
SubstituteChar() : Util::String
SubstituteString() : Util::String
SupportsDisplayMode() : Direct3D9::D3D9DisplayDevice, Base::DisplayDeviceBase, AsyncGraphics::DisplayProxy
SwitchFocusEntities() : BaseGameFeature::FocusManager
SystemTimeToFileTime() : Win32::Win32CalendarTime, Base::CalendarTimeBase
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
- t -

- TcpClient() : Net::TcpClient
- TcpClientConnection() : Net::TcpClientConnection
- TcpServer() : Net::TcpServer
- Tell() : Win32::Win32FSWrapper
- TerminateAtIndex() : Util::String
- Text() : Http::HtmlPageWriter
- TextureBase() : Base::TextureBase
- TexturePageHandler() : Debug::TexturePageHandler
- ThreadSafeDisplayEventHandler() : CoreGraphics::ThreadSafeDisplayEventHandler
- ThreadSafeRenderEventHandler() : CoreGraphics::ThreadSafeRenderEventHandler
- ThreadStopRequested() : Win32::Win32Thread
- to_matrix44() : Math::bbox
- ToHtml() : Http::HtmlElement
- ToHumanReadableString() : Http::HttpStatus
- Tokenize() : Util::String
- ToLower() : Util::String
- ToMediaType() : CoreGraphics::ImageFileFormat
- ToString() : Input::Key, Input::MouseButton, CoreGraphics::PixelFormat, Models::ModelNodeType, Http::HttpMethod, Http::HttpStatus, Util::FourCC, CoreGraphics::Adapter, CoreGraphics::AntiAliasQuality, CoreGraphics::BatchType, CoreGraphics::ImageFileFormat, CoreGraphics::IndexType, CoreGraphics::PrimitiveTopology, Frame::LightingMode, Frame::SortingMode
- ToUpper() : Util::String
- transform() : **Math::bbox**
- transform44() : **Math::transform44**
- TransformDevice() : **CoreGraphics::TransformDevice**
- TransformDeviceBase() : **Base::TransformDeviceBase**
- TransformNode() : **Models::TransformNode**
- TransformNodeInstance() : **Models::TransformNodeInstance**
- TranslateKeyCode() : **Win32::Win32DisplayDevice**
- Trim() : **Util::String**
- TrimLeft() : **Util::String**
- TrimRight() : **Util::String**
- TypeToString() : **Base::ShaderVariableBase**, **Util::Variant**
- u -

- Unload() : Base::MeshBase, Base::VertexBufferBase, CoreGraphics::CPUVertexBuffer, Direct3D9::D3D9VertexBuffer, Models::Model, Direct3D9::D3D9IndexBuffer, Resources::Resource, CoreGraphics::CPUIndexBuffer, Direct3D9::D3D9Shader, Direct3D9::D3D9Texture
- UnloadResources() : Models::StateNode, Models::Model, Models::ModelNode, Models::CharacterNode, Models::ParticleSystemNode, Models::ShapeNode
- Unmap() : IO::FileStream, IO::MemoryStream, IO::Stream, IO::ZipFileStream, Base::IndexBufferBase, Base::TextureBase, Base::VertexBufferBase, CoreGraphics::CPUIndexBuffer, CoreGraphics::CPUVertexBuffer, Direct3D9::D3D9IndexBuffer, Direct3D9::D3D9Texture, Direct3D9::D3D9VertexBuffer
- UnmapCubeFace() : Base::TextureBase, Direct3D9::D3D9Texture
- Unmount() : IO::ZipFileSystem
- UnmountZipArchive() : IO::IoServer
- Unregister() : Attr::AttributeDefinitionBase
- UnregisterCommand() : Scripting::LuaServer, Scripting::ScriptServer
- UnregisterSharedResource() : Resources::SharedResourceServer
- UnregisterUriScheme() : IO::IoServer
- Update() : Models::ModelNodeInstance, IO::Console
IO::ConsoleHandler, Models::ModelInstance, Models::CharacterNodeInstance, Models::ParticleSystemNodeInstance, Models::SkinShapeNodeInstance, RenderUtil::MayaCameraUtil, Resources::ResourceManager, Models::TransformNodeInstance

- UpdateButtonState(): XInput::XInputGamePad
- UpdateCameraLinks(): Graphics::Stage
- UpdateEntities(): Graphics::Stage
- UpdateGlobalBoundingBox(): Graphics::GraphicsEntity
- UpdateLightLinks(): Graphics::Stage
- UpdateLinks(): Graphics::Cell
- UpdateManagedTextureVariables(): Models::StateNode
- UpdatePositionInCellTree(): Graphics::GraphicsEntity
- UpdateProgressDisplay(): BaseGameFeature::LoaderServer
- UpdateRenderStats(): Resources::ManagedResource
- UpdateShadowBuffers(): Lighting::SM30ShadowServer
- UpdateThumbAxis(): XInput::XInputGamePad
- UpdateTriggerAxis(): XInput::XInputGamePad
- UpdateTriggeredEntities(): BaseGameFeature::EntityManager
- UpdateViewProjMatrix(): Graphics::CameraEntity, AsyncGraphics::CameraEntityProxy
- UpdateVisibilityLinks(): Graphics::View
- upvec(): Math::vector
- URI(): IO::URI
- UserInfo(): IO::URI
- UserProfile(): BaseGameFeature::UserProfile
- UTF8toANSI(): Util::String
Main Page
Namespaces
Data Structures
Files
Related Pages

Alphabetical List
Data Structures
Class Hierarchy
Data Fields

All
Functions
Variables
Typedefs
Enumerations
Related Functions

a
b
c
d
e
f
g
h
i
k
l
m
n
o
p
q
r
s
t
- v -

- ValidateCharacter() : Models::CharacterNodeInstance, Models::SkinShapeNodeInstance
- ValidateEntityHandle() : AsyncGraphics::GraphicsEntityProxy
- ValidateModelInstance() : Graphics::ModelEntity
- Value() : Util::SimpleTree< VALUETYPE >::Node, Util::Atom< TYPE >, Util::KeyValuePair< KEYTYPE, VALUETYPE >
- ValueAsString() : Attr::Attribute
- ValueAtAtIndex() : Util::Dictionary< KEYTYPE, VALUETYPE >
- Values() : BaseGameFeature::CategoryManager::Entry
- ValuesAsArray() : Util::Dictionary< KEYTYPE, VALUETYPE >
- ValueTypeToString() : Attr::Attribute
- Variant() : Util::Variant
- vec() : Math::line
- vector() : Math::vector
- VertexBufferBase() : Base::VertexBufferBase
- VertexComponent() : CoreGraphics::VertexComponent
- VertexLayoutBase() : Base::VertexLayoutBase
- VertexLayoutServer() : CoreGraphics::VertexLayoutServer
- View() : Graphics::View
- ViewerApplication() : App::ViewerApplication
- ViewProxy() : AsyncGraphics::ViewProxy
- VisResolveContainer() : Models::VisResolveContainer< TYPE >
- VisResolver() : Models::VisResolver
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
- W -

- Wait() : Messaging::AsyncPort, Threading::SafePriorityQueue< PRITYPE, TYPE >, Win32::Win32Event, Threading::SafeQueue< TYPE >
- WaitTimeout() : Win32::Win32Event
- Warning() : IO::ConsoleHandler, Win32::Win32ConsoleHandler, IO::Console
- WeekdayToString() : Base::CalendarTimeBase
- WheelBackward() : Base::MouseBase
- WheelForward() : Base::MouseBase
- width() : Math::rectangle< TYPE >
- Width() : Util::FixedTable< TYPE >
- Win32ConsoleHandler() : Win32::Win32ConsoleHandler
- Win32CriticalSection() : Win32::Win32CriticalSection
- Win32DisplayDevice() : Win32::Win32DisplayDevice
- Win32Event() : Win32::Win32Event
- Win32FileTime() : Win32::Win32FileTime
- Win32Guid() : Win32::Win32Guid
- Win32Heap() : Win32::Win32Heap
- Win32InputServer() : Win32::Win32InputServer
- Win32IpAddress() : Win32::Win32IpAddress
- Win32Socket() : Win32::Win32Socket
- Win32Thread() : Win32::Win32Thread
- Win32Timer() : Win32::Win32Timer
- WinProc() : Win32::Win32DisplayDevice
- Write() : Http::Base64Writer, IO::FileStream, IO::MemoryStream, IO::Stream, Win32::Win32FSWrapper, System::Win32Registry
- WriteBlob() : IO::BinaryWriter
- WriteBool() : IO::BinaryWriter
- WriteChar() : IO::BinaryWriter, IO::TextWriter
- WriteContent() : IO::XmlWriter
- WriteDouble() : IO::BinaryWriter
- WriteFloat() : IO::BinaryWriter
- WriteFloat4() : 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
- WriteModelAttrs() : Models::BinaryModelWriter, Models::ModelWriter, Models::XmlModelWriter
- WriteModelNodeAttrs() : Models::BinaryModelWriter, Models::XmlModelWriter, Models::ModelWriter
- WriteResponse() : Http::HttpResponseWriter
- WriteShort() : IO::BinaryWriter
- WriteString() : IO::TextWriter, IO::BinaryWriter
- WriteUChar() : IO::BinaryWriter
- WriteUInt() : IO::BinaryWriter
- WriteUShort() : IO::BinaryWriter
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
- X -

- x() : Math::float2
- XInputGamePad() : XInput::XInputGamePad
- XmlModelReader() : Models::XmlModelReader
- XmlModelWriter() : Models::XmlModelWriter
- XmlReader() : IO::XmlReader
- XmlWriter() : IO::XmlWriter

The Nebula Device 3 documentation generated by doxygen at Tue Feb 19 12:16:40 2008
- Main Page
- Namespaces
- Data Structures
- Files
- Related Pages

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

- All
- Functions
- Variables
- Typedefs
- Enumerations
- Related Functions

- a
- b
- c
- d
- e
- f
- g
- h
- i
- k
- l
- m
- n
- o
- p
- q
- r
- s
- t
- y -

- y() : Math::float2
- Z -

- ZipArchive() : IO::ZipArchive
- ZipDirEntry() : IO::ZipDirEntry
- ZipFileEntry() : IO::ZipFileEntry
- ZipFileStream() : IO::ZipFileStream
- ZipFileSystem() : IO::ZipFileSystem
- AbstractLightEntity() : Lighting::AbstractLightEntity
- ActorEntity() : Graphics::ActorEntity
- Application() : App::Application
- ArgsBlock() : Scripting::ArgsBlock
- Array() : Util::Array<TYPE>
- AsyncGraphicsHandler() : AsyncGraphics::AsyncGraphicsHandler
- AsyncGraphicsInterface() : AsyncGraphics::AsyncGraphicsInterface
- AsyncHttpInterface() : AsyncHttp::AsyncHttpInterface
- AsyncPort() : Messaging::AsyncPort
- AsyncRenderApplication() : App::AsyncRenderApplication
- AsyncViewerApplication() : App::AsyncViewerApplication
- Atom() : Util::Atom<TYPE>
- AttributeContainer() : Attr::AttributeContainer
- AttributeDefinitionBase() : Attr::AttributeDefinitionBase
- AttributeTable() : Attr::AttributeTable
- Base64Writer() : Http::Base64Writer
- BinaryModelReader() : Models::BinaryModelReader
- BinaryModelWriter() : Models::BinaryModelWriter
- BinaryReader() : IO::BinaryReader
- BinaryWriter() : IO::BinaryWriter
- Blob() : Util::Blob
- CameraEntity() : Graphics::CameraEntity
- CameraEntityProxy() : AsyncGraphics::CameraEntityProxy
- CategoryManager() : BaseGameFeature::CategoryManager
- Cell() : Graphics::Cell
- ~CharacterNode() : Models::CharacterNode
- ~CharacterNodeInstance() : Models::CharacterNodeInstance
- ~Command() : Scripting::Command
- ~Console() : IO::Console
- ~ConsoleApplication() : App::ConsoleApplication
- ~ConsoleHandler() : IO::ConsoleHandler
- ~CoreServer() : Core::CoreServer
- ~CPUIndexBuffer() : CoreGraphics::CPUIndexBuffer
- ~CPUVertexBuffer() : CoreGraphics::CPUVertexBuffer
- ~D3D9DisplayDevice() : Direct3D9::D3D9DisplayDevice
- ~D3D9IndexBuffer() : Direct3D9::D3D9IndexBuffer
- ~D3D9RenderDevice() : Direct3D9::D3D9RenderDevice
- ~D3D9RenderTarget() : Direct3D9::D3D9RenderTarget
- ~D3D9Shader() : Direct3D9::D3D9Shader
- ~D3D9ShaderInstance() : Direct3D9::D3D9ShaderInstance
- ~D3D9ShaderServer() : Direct3D9::D3D9ShaderServer
- ~D3D9ShaderVariable() : Direct3D9::D3D9ShaderVariable
- ~D3D9ShaderVariation() : Direct3D9::D3D9ShaderVariation
- ~D3D9ShapeRenderer() : Direct3D9::D3D9ShapeRenderer
- ~D3D9Texture() : Direct3D9::D3D9Texture
- ~D3D9VertexBuffer() : Direct3D9::D3D9VertexBuffer
- ~D3D9VertexLayout() : Direct3D9::D3D9VertexLayout
- ~DisplayDevice() : CoreGraphics::DisplayDevice
- ~DisplayDeviceBase() : Base::DisplayDeviceBase
- ~DisplayEventHandler() : CoreGraphics::DisplayEventHandler
- ~DisplayProxy() : AsyncGraphics::DisplayProxy
- ~DynamicMeshResourceLoader() : Resources::DynamicMeshResourceLoader
- ~Entity() : Game::Entity
- ~EntityLoaderBase() : BaseGameFeature::EntityLoaderBase
- ~EntityManager() : BaseGameFeature::EntityManager
- ~EnvEntityManager() : BaseGameFeature::EnvEntityManager
- ~EnvQueryManager() : BaseGameFeature::EnvQueryManager
- ~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
~FramePostEffect() : Frame::FramePostEffect
~FrameServer() : Frame::FrameServer
~FrameShader() : Frame::FrameShader
~GameApplication() : App::GameApplication
~GamePadBase() : Base::GamePadBase
~GameServer() : Game::GameServer
~GameStateHandler() : App::GameStateHandler
~GlobalAttrsManager() :
    BaseGameFeature::GlobalAttrsManager
~GraphicsEntity() : Graphics::GraphicsEntity
~GraphicsEntityProxy() : AsyncGraphics::GraphicsEntityProxy
~GraphicsServer() : Graphics::GraphicsServer
~GraphicsServerProxy() :
    AsyncGraphics::GraphicsServerProxy
~Handler() : Messaging::Handler
~HtmlPageWriter() : Http::HtmlPageWriter
~HttpRequest() : Http::HttpRequest
~HttpRequestHandler() : Http::HttpRequestHandler
~HttpServer() : Http::HttpServer
~IndexBufferBase() : Base::IndexBufferBase
~InputHandler() : Input::InputHandler
~InputServer() : Input::InputServer
~InputServerBase() : Base::InputServerBase
~IoServer() : IO::IoServer
~KeyboardBase() : Base::KeyboardBase
~LightServer() : Lighting::LightServer
~LightServerBase() : Lighting::LightServerBase
~List() : Util::List<TYPE>
~LoaderServer() : BaseGameFeature::LoaderServer
~LuaServer() : Scripting::LuaServer
~ManagedResource() : Resources::ManagedResource
~Manager() : Game::Manager
~MemoryStream() : IO::MemoryStream
~MeshBase() : Base::MeshBase
~Model() : Models::Model
~ModelEntity() : Graphics::ModelEntity
~ModelInstance() : Models::ModelInstance
- `~ModelNode()`: `Models::ModelNode`
- `~ModelNodeInstance()`: `Models::ModelNodeInstance`
- `~ModelReader()`: `Models::ModelReader`
- `~ModelServer()`: `Models::ModelServer`
- `~ModelWriter()`: `Models::ModelWriter`
- `~MouseBase()`: `Base::MouseBase`
- `~MouseGripperProperty()`: `PhysicsFeature::MouseGripperProperty`
- `~N2ModelReader()`: `Models::N2ModelReader`
- `~Node()`: `Util::QuadTree< TYPE >::Node`, `Util::SimpleTree< VALUETYPE >::Node`
- `~ParticleSystemNode()`: `Models::ParticleSystemNode`
- `~ParticleSystemNodeInstance()`: `Models::ParticleSystemNodeInstance`
- `~PhysicsProperty()`: `PhysicsFeature::PhysicsProperty`
- `~Property()`: `Game::Property`
- `~QuadtreeStageBuilder()`: `Graphics::QuadtreeStageBuilder`
- `~RefCounted()`: `Core::RefCounted`
- `~RenderApplication()`: `App::RenderApplication`
- `~RenderDevice()`: `CoreGraphics::RenderDevice`
- `~RenderDeviceBase()`: `Base::RenderDeviceBase`
- `~RenderEventHandler()`: `CoreGraphics::RenderEventHandler`
- `~RenderTargetBase()`: `Base::RenderTargetBase`
- `~Resource()`: `Resources::Resource`
- `~ResourceBase()`: `Base::ResourceBase`
- `~ResourceLoader()`: `Resources::ResourceLoader`
- `~ResourceManager()`: `Resources::ResourceManager`
- `~ResourceMapper()`: `Resources::ResourceMapper`
- `~ResourceSaver()`: `Resources::ResourceSaver`
- `~ScriptServer()`: `Scripting::ScriptServer`
- `~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
- SharedResourceServer() : Resources::SharedResourceServer
- SimpleResourceMapper() : Resources::SimpleResourceMapper
- SimpleStageBuilder() : Graphics::SimpleStageBuilder
- SkinShapeNode() : Models::SkinShapeNode
- SkinShapeNodeInstance() : Models::SkinShapeNodeInstance
- SM30LightServer() : Lighting::SM30LightServer
- SM30ShadowServer() : Lighting::SM30ShadowServer
- Stage() : Graphics::Stage
- StageBuilder() : Graphics::StageBuilder
- StateNode() : Models::StateNode
- StateNodeInstance() : Models::StateNodeInstance
- Stream() : IO::Stream
- StreamAnimationLoader() : CoreGraphics::StreamAnimationLoader
- StreamMeshLoader() : CoreGraphics::StreamMeshLoader
- StreamModelLoader() : Models::StreamModelLoader
- StreamReader() : IO::StreamReader
- StreamTextureSaverBase() : Base::StreamTextureSaverBase
- StreamWriter() : IO::StreamWriter
- String() : Util::String
- TcpClient() : Net::TcpClient
- TcpClientConnection() : Net::TcpClientConnection
- TcpServer() : Net::TcpServer
- TextureBase() : Base::TextureBase
- ThreadSafeDisplayEventHandler() : CoreGraphics::ThreadSafeDisplayEventHandler
- ThreadSafeRenderEventHandler() : CoreGraphics::ThreadSafeRenderEventHandler
- TransformDevice() : CoreGraphics::TransformDevice
- TransformDeviceBase() : Base::TransformDeviceBase
- TransformNode() : Models::TransformNode
- TransformNodeInstance() : Models::TransformNodeInstance
- UserProfile() : BaseGameFeature::UserProfile
- Variant() : Util::Variant
- VertexBufferBase() : Base::VertexBufferBase
- VertexLayoutBase() : Base::VertexLayoutBase
- VertexLayoutServer() : CoreGraphics::VertexLayoutServer
- View() : Graphics::View
- ViewerApplication() : App::ViewerApplication
- ViewProxy() : AsyncGraphics::ViewProxy
- VisResolver() : Models::VisResolver
- Win32CriticalSection() : Win32::Win32CriticalSection
- Win32DisplayDevice() : Win32::Win32DisplayDevice
- Win32Event() : Win32::Win32Event
- Win32Heap() : Win32::Win32Heap
- Win32InputServer() : Win32::Win32InputServer
- Win32Socket() : Win32::Win32Socket
- Win32Thread() : Win32::Win32Thread
- XInputGamePad() : XInput::XInputGamePad
- XmlModelReader() : Models::XmlModelReader
- XmlModelWriter() : Models::XmlModelWriter
- XmlReader() : IO::XmlReader
- XmlWriter() : IO::XmlWriter
- ZipArchive() : IO::ZipArchive
- ZipFileEntry() : IO::ZipFileEntry
- ZipFileStream() : IO::ZipFileStream
- ZipFileSystem() : IO::ZipFileSystem