

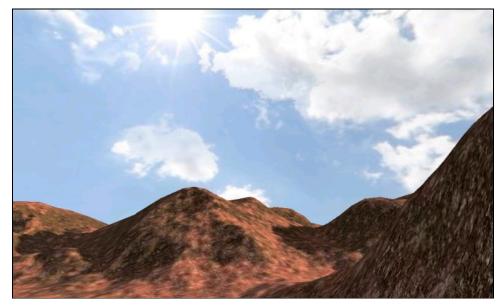
2.04 (2010-10-09) ©2010 - see <u>Credits</u> au3lrr2 Project page

Introduction

au3Irr2 brings together the ease of <u>Autolt</u> with the power of the 3D engine **Irrlicht**. It covers a whole bunch of features to create serious 3D applications as well as nice games, useable both by beginners and more experienced to learn, prototye, and realise ideas without a big overhead.

Also au3Irr2 is still work in progress, it already provides well over 260 commands that cover features like Bitmaps, 3D models, Animation, Collision, Scene management, Maps, Terrains, Cameras, Lights, special effects, GUI and more. Additionally, there are around 100 well commented examples showing the different features and their usage.

Technically, au3Irr2 is a 'wrapper for a wrapper', based upon Frank Dodd's great **FreeBasic Irrlicht Wrapper**. Standing on this mature project au3Irr2 has really good potential to grow and prosper. Especially if you use it and be active in the <u>forum</u>!



How to start

Pretty easy. Of course you need installed <u>Autolt</u>, also full version of <u>SciTE</u> is highly recommend.

- Extract the au3Irr2.zip to whereever you like.
- Run the 'example launcher' to run & enjoy the **examples**. Open them via the launcher with SciTE to see how it is done.
- Use the **helpfile** it lists all *working* functions, so it can be used as reference and overview.
- Use the **setup** feature inside the example launcher to add **context help and calltips** into SciTE.

Remark

Because helpfile does (yet) not provide detailed informations about parameters and usage of all working functions, documentation of the original Freebasic Wrapper is also included as appendix.

au3Irr2 follows (mostly) its syntax and usage, so it's a good place to look into (keep in mind: not all FB Wrapper functions are also implemented in au3Irr2).



Where to continue

- jRowe's au3Irr2 topic on autoitscript.com. Don't be shy, be part of it.
- <u>au3Irr2 project page</u> with latest releases and sources.
- Frank Dodd's Irrlicht Wrapper Online Portal.
- Frank Dodd's Irrlicht Wrapper in the FreeBasic Forum.
- Irrlicht engine the 'mother' of it all.

Software License

au3lrr2

WWW : http://code.google.com/p/au3irrlicht2/ Contact : via au3Irr2 topic on autoitscript.com (http://www.autoitscript.com/forum/index.php?showtopic=113881) Authors : J.Rowe and Andreas Templin

END-USER LICENSE AGREEMENT FOR THIS SOFTWARE

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

ADDENDUM

au3Irr2 is based upon the open source projects

- 'IrrlichtWrapper for FreeBasic' from Frank Dodd and the IrrlichtWrapper for FB team (http://www.freebasic.net/forum/viewtopic.php?t=3584), and
- 'Irrlicht Engine' from Nikolaus Gebhardt and the Irrlicht team (http://irrlicht.sourceforge.net/).
 The Irrlicht Engine itself is based in part on the work of the Independent JPEG Group and the zlib.

Please refer to documentations and license agreements of this two projects for further information.

[END OF LICENSE]

au3lrr2 History/Changelog

This file was created automatically as a subset. See **\include_au3Irr2_changelog.txt** for complete list of changes.

Release 2.04 (2010-10-09)

Script breaking changes:

- _IrrAddSplineAnimator: Parameters changed to simplify usage (see examples)
- _IrrGetNodePosition, _IrrGetNodeRotation, _IrrGetNodeAbsolutePosition, _IrrGetCameraTarget: Removed required byRef array parameter (see help file/adjusted examples for syntax)

UDFs:

- Fixed/Added: Some more working functions (mainly around added examples)
- Added: Some missing 2D functions and documentation (contributions from smashly)
- Fixed: _IrrSetFog (linear and exponential fog were interchanged)
- Fixed: _IrrCreateMesh (\$s_MeshName not being passed in the DLL call)
- Fixed: _IrrGetScreenCoordinatesFrom3DPosition (did not return anything useable because of wrong dllCall)

Examples:

- Added: 015 (CustomMesh), 022 (Indices+Vertices), 024 (Mesh to file), 062 (6D0F_Camera)
- Added: (contribution from smashly): 039 (Texture blending), 070 (Texture_and_Images), 103 (Billboard Groups), 104 (LOD)
- Added: 007/025/027/066/067/068/069/074/075 (ParticleSystem)
- Reworked: 011 (Animators) shows usage of changed _IrrAddSplineAnimator
- Fixed: 010 (TerrainAndFog) / 029/054 (Skydomes)

Help file:

 Added: Completed documentation including example code for some more functions. Current status: Topics completed along examples 1-16 + several more. At least naked reference w/o detailed informations for all functions inside other examples.

• Added: 'Copy to clipboard'-button for included examples (using VBS code from GEOSoft)

Internal tools:

- Fixed: Examples could not be opened when running launcher from path with spaces
- Added: Setup feature to example launcher to merge au3Irr2 help into local au3 help and add/update calltips for SciTe
- Excluded \internal_tools because not too interesting for 99.9% (for the 0.1%: available via the sources from project page)

Other:

• Changed: global \$result used in UDF files switched to local variables (WIP, finished for 2D, Scene, Camera, Animation, Node)

Release 2.03 (2010-09-05)

UDF:

- Changed: Splitted UDF per category into \include. Main UDF (and the only one which is needed to be included) is still au3Irrlicht2.au3
- Changed: All used dll files are moved to \bin to clean up the root dir. _IrrStart is modified to find them anyway.
- Fixed: _IrrSetNodeVisibility

Examples:

- Added: 089 (Orthogonal Camera), 106 (Advanced start)
- Changed: 029 (Skydome)
- Fixed: 049 (Loaded_Scene_Collision), 051 (Clouds)

Help file:

- Added: First version as reference of all working + proven functions. Current status: Topics completed including example code along examples 1 to 6. Naked reference w/o detailed informations for other examples.
- Changed: Moved original freeBasic Wrapper docs from \FB_documentation to \internal_tools\buildHelp\html_static. It's now included into help file.
- Added: Merged help to use the au3Irr2 help inside the au3 help.

Internal tools:

 Added: helper scripts and files for building help file and au3.user.calltips.api (see \internal tools\help_building_readme.txt)

Other:

- Fixed: \media\fonthaettenschweiler.bmp (bogQ)
- Added: msvcr71.dll possibly missing on some machines (jl)

Release 2.02 (2010-07-29)

First release in one package (updated UDF + all needed .dll's, more examples, ExampleLauncher)

Release 2.01 (2010-07-20)

Updated UDF with fixes, additions and separate example package

Release 2.00 (2010-05-03)

First release of jRowe

Tutorials

Where are the tutorials?

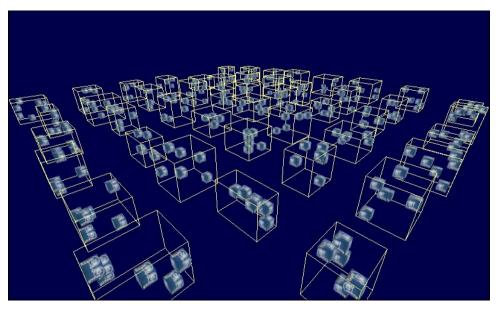
There are no tutorials nor bigger demos yet. But why not be the first to write one?

Get yourself a place on the <u>credits page</u> and help **au3Irr2** to become a mature project!

In the meantime ...

... you are not alone. There are a lot of well commented examples in the **\example directory**. They should give you all you need to work yourself into **au3Irr2**.

Have fun!



IrrSetTextureCreationFlag

Sets texture creation flags controlling how textures are handled when they are created.

#Include <au3Irrlicht2.au3>
_IrrSetTextureCreationFlag(\$i_Flag, \$i_Value)

Parameters

\$i_Value	The following flags can be set; \$ETCF_ALWAYS_16_BIT - Forces the driver to always create 16 bit textures, independently of which format the file on disk has. \$ETCF_ALWAYS_32_BIT - Forces the driver to always create 32 bit textures, independently of which format the file on disk has. \$ETCF_OPTIMIZED_FOR_QUALITY - Lets the driver decide in which format the textures are created and tries to make the textures look as good as possible. \$ETCF_OPTIMIZED_FOR_SPEED - Lets the driver decide in which format the textures are created and tries to create them maximizing render speed. \$ETCF_CREATE_MIP_MAPS - Automatically creates mip map levels for the textures. \$ETCF_NO_ALPHA_CHANNEL - Discard any alpha layer and use non-alpha color format.
\$i_Flag	Turn Creation Flag Off or On (\$IRR_OFF or \$IRR_ON)

Return Value

Success: True Failure: False

Remarks

None.

Related

None.

Example

#include <au3Irrlicht2.au3>
Global \$hTexture, \$aInfo

_IrrStart()

; Set the Texture creation flag to load textures in 16 bit without alpha (R5G6B5 format)

_IrrSetTextureCreationFlag(*BitOR*(\$ETCF_ALWAYS_16_BIT, \$ETCF_NO_ALPHA_CHANNEL), \$IRR_ON)

; Load a texture
\$hTexture = _IrrGetTexture("./media/cross.bmp")

; query some info about the loaded texture, index 3 of the returned array is Color Reference

\$aInfo = _IrrGetTextureInformation(\$hTexture)

; show the color format in a string to see the Texture creation flag was set *MsgBox*(64, "Texture color format", _TextureFormatString(\$aInfo[3]))

_IrrStop()

Func _TextureFormatString(\$iValue)

```
Local $sMsg
  Switch $iValue
    Case $ECF R5G6B5
      $sMsg &= "R5G6B5 - 16 bit without alpha channel"
    Case $ECF_A1R5G5B5
      $sMsg &= "A1R5G5B5 - 16 bit with alpha channel"
    Case $ECF_R8G8B8
      $sMsg &= "R8G8B8 - 24 bit without alpha channel"
    Case $ECF A8R8G8B8
      $sMsg &= "A8R8G8B8 - 32 bit with alpha channel"
    Case Else
      $sMsg &= "Unknown"
  EndSwitch
  Return $sMsg
EndFunc ;==>_TextureFormatString
Copy to Clipboard
```

IrrGetTexture

Loads 2D texture from bitmap file into video memory that can then be used to texture a model or to draw onto the screen.

#Include <au3Irrlicht2.au3>
_IrrGetTexture(\$s_ImageFile)

Parameters

\$s_ImageFile Full path to the bitmap file.

Return Value

Success: Handle of the device dependend irrlicht texture object Failure: False

Remarks

Irrlicht engine supports currently this image file formats:

JPEG File Interchange Format (.jpg, r/w)	Portable Network Graphics (.png, r/w)
/ / · · · · · · · · · · · · · · · · · ·	Windows Bitmap (.bmp, r/w)
Zsoft Paintbrush (.pcx, r/w)	Portable Pixmaps (.ppm, r/w)
Adobe Photoshop (.psd, r)	Quake 2 textures (.wal, r)

Related

_IrrRemoveTexture, _IrrGetImage

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $camera = _IrrAddCamera(2,2,2, 0,0,0 )
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $sceneNode = _IrrAddMeshToScene( $mesh )
local $texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
WHILE _IrrRunning()
  _IrrBeginScene(50, 50, 50)
  _IrrDrawScene()
  _IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrGetImage

Loads 2D texture from bitmap file into main memory for CPU based operations.

#Include <au3Irrlicht2.au3>
_IrrGetImage(\$s_ImageFile)

Parameters

\$s_ImageFile Full path to the bitmap file.

Return Value

Success: Handle of the irrlicht memory texture object Failure: False

Remarks

This images can not be used to texture 3D objects! Instead, they can be used to supply a heightmap to a terrain or other similar CPU based operations.

Irrlicht engine supports currently this image file formats:

JPEG File Interchange Format (.jpg,	
r/w)	r/w)
Truevision Targa (.tga, r/w)	Windows Bitmap (.bmp, r/w)
Zsoft Paintbrush (.pcx, r/w)	Portable Pixmaps (.ppm, r/w)
Adobe Photoshop (.psd, r)	Quake 2 textures (.wal, r)

Related

<u>_IrrRemoveImage</u>, <u>_IrrGetTexture</u>

IrrCreateTexture

Creates a blank texture.

#Include <au3Irrlicht2.au3>
_IrrCreateTexture(\$s_TextureName, \$i_XSize, \$i_YSize, \$i_ColorFormat)

Parameters

\$s_TextureName	Texture_name as string.
\$i_XSize	Width of the texture.
\$i_YSize	Height of the texture.
\$i_ColorFormat	The format of the texture can be one of the following: \$ECF_A1R5G5B5 - 16 bit color format used by the software driver, and thus preferred by all other irrlicht engine video drivers. \$ECF_R5G6B5 - Standard 16 bit color format. \$ECF_R8G8B8 - 24 bit color, no alpha channel, but 8 bit for red, green and blue. \$ECF_A8R8G8B8 - Default 32 bit color format. 8 bits are used for every component: red, green, blue and alpha.

Return Value

Success: Handle of the newly created irrlicht texture object Failure: False and @error 1

Remarks

None.

Related

_IrrDraw2DImage, _IrrGetTextureInformation, _IrrLockTexture, _IrrUnlockTexture

Example

```
#include "au3Irrlicht2.au3"
Global $hTexture
_IrrStart()
$hTexture = _IrrCreateTexture("MyTexture", 128, 128, $ECF_A8R8G8B8)
While _IrrRunning()
_IrrBeginScene(255, 255, 0)
```

```
_IrrBraw2DImage($hTexture, 0, 0)
_IrrEndScene()
```

WEnd

```
_IrrStop()
```

Copy to Clipboard

IrrCreateImage

Creates a blank image that does not use video memory.

#Include <au3Irrlicht2.au3>
_IrrCreateImage(\$i_XSize, \$i_YSize, \$i_ColorFormat)

Parameters

\$i_XSize	Width of the texture.
\$i_XSize	Width of the texture.
\$i_ColorFormat	The format of the texture can be one of the following: \$ECF_A1R5G5B5 - 16 bit color format used by the software driver, and thus preferred by all other irrlicht engine video drivers. \$ECF_R5G6B5 - Standard 16 bit color format. \$ECF_R8G8B8 - 24 bit color, no alpha channel, but 8 bit for red, green and blue. \$ECF_A8R8G8B8 - Default 32 bit color format. 8 bits are used for every component: red, green, blue and alpha.

Return Value

Success: Handle of the newly created device dependend irrlicht image object

Failure: False and @error 1

Remarks

This images can not be used to texture 3D objects! Instead, they can be used to supply a heightmap to a terrain or other similar CPU based operations.

Related

_IrrLockImage, _IrrUnlockImage

IrrRemoveTexture

Removes the texture from memory freeing up the space it occupied.

#Include <au3Irrlicht2.au3>
_IrrRemoveTexture(\$h_Texture)

Parameters

\$h_Texture Handle of an device dependend irrlicht texture object

Return Value

Success: True Failure: False

Remarks

You should ensure that the texture is not in use by materials assigned to nodes.

Related

<u>_IrrGetTexture</u>, <u>_IrrGetImage</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $camera = _IrrAddCamera(2,2,2, 0,0,0 )
```

```
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $sceneNode = _IrrAddMeshToScene( $mesh )
```

```
local $texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
```

```
local $time = TimerInit()
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
```

```
if $sceneNode <> 0 AND TimerDiff($time) > 3000 then
    _IrrRemoveNode($sceneNode)
    _IrrRemoveTexture($texture) ; no longer needed
    $sceneNode = 0
```

EndIf

```
WEND
```

_IrrStop()

```
Copy to Clipboard
```

IrrRemoveImage

Removes the image from memory freeing up the space it occupied.

#Include <au3Irrlicht2.au3>
_IrrRemoveImage(\$h_Image)

Parameters

\$h_Image Handle of the irrlicht memory image object.

Return Value

Success: True Failure: False

Remarks

You should ensure that the image is not in use by other functions.

Related

_IrrGetImage, _IrrGetTexture

IrrLockTexture

Locks the texture and returns a pointer to the pixels.

#Include <au3Irrlicht2.au3>
_IrrLockTexture(\$h_Texture)

Parameters

\$h_Texture Handle to an irrlicht texture object

Return Value

Success: Pointer to the pixels. Failure: False and @error 1

Remarks

None.

Related

_IrrUnlockTexture

Example

#include "au3Irrlicht2.au3"

Global \$hTexture Global \$iWidthHeight = 128

```
Global $iPixelsAmount = $iWidthHeight * $iWidthHeight
Global $pPixels, $tPixels
Global $iColor = 0xFFFF0000 ; Red
```

_IrrStart()

```
$hTexture = _IrrCreateTexture("Red", $iWidthHeight, $iWidthHeight,
$ECF_A8R8G8B8)
$pPixels = _IrrLockTexture($hTexture)
$tPixels = DllStructCreate("uint[" & $iPixelsAmount & "]", $pPixels)
For $i = 1 To $iPixelsAmount
  DllStructSetData($tPixels, 1, $iColor, $i)
  $pPixels += 1
Next
_IrrUnlockTexture($hTexture)
tPixels = 0
While _IrrRunning() And Sleep(10)
  _IrrBeginScene(255, 255, 0)
  _IrrDraw2DImage($hTexture, 0, 0)
  _IrrEndScene()
WEnd
_IrrStop()
Copy to Clipboard
```

IrrUnlockTexture

Unlock the texture, presumably after it has been modified and recreate the mipmap levels.

#Include <au3Irrlicht2.au3>
_IrrUnlockTexture(\$h_Texture)

Parameters

\$h_Texture Handle to an irrlicht texture object that has been Locked by prior call to _IrrLockOpenGLTexture or _IrrLockTexture.

Return Value

Success: True Failure: False

Remarks

None

Related

_IrrLockOpenGLTexture, _IrrLockTexture

Example

#include "au3Irrlicht2.au3"

```
Global $hTexture
Global $iWidthHeight = 128
Global $iPixelsAmount = $iWidthHeight * $iWidthHeight
Global $pPixels, $tPixels
Global $iColor = 0xFFFF0000 ; Red
```

_IrrStart()

```
$hTexture = IrrCreateTexture("Red", $iWidthHeight, $iWidthHeight,
$ECF_A8R8G8B8)
$pPixels = _IrrLockTexture($hTexture)
$tPixels = DllStructCreate("uint[" & $iPixelsAmount & "]", $pPixels)
For $i = 1 To $iPixelsAmount
  DllStructSetData($tPixels, 1, $iColor, $i)
  $pPixels += 1
Next
_IrrUnlockTexture($hTexture)
tPixels = 0
While _IrrRunning() And Sleep(10)
  _IrrBeginScene(255, 255, 0)
  _IrrDraw2DImage($hTexture, 0, 0)
  _IrrEndScene()
WEnd
_IrrStop()
Copy to Clipboard
```

IrrLockImage

Locks an image object and returns a pointer to the pixels.

#Include <au3Irrlicht2.au3>
_IrrLockImage(\$h_Image)

Parameters

\$h_Image Handle to an irrlicht image object

Return Value

Success: Pointer to the image pixels Failure: False and @error 1

Remarks

None.

Related

_IrrCreateImage, _IrrGetImage, _IrrUnlockImage

Example

#include "au3Irrlicht2.au3"

Global \$hSrcImage, \$hDesTexture Global \$pSrcPixels, \$pDesPixels

```
Global $tSrcPixels, $tDesPixels
Global $iWidthHeight = 256
Global $iPixelsAmount = $iWidthHeight * $iWidthHeight
```

_IrrStart()

```
$hSrcImage = _IrrGetImage("./media/splatter.tga")
$hDesTexture = _IrrCreateTexture("Desination", $iWidthHeight,
$iWidthHeight, $ECF_A8R8G8B8)
```

```
$pSrcPixels = _IrrLockImage($hSrcImage)
$pDesPixels = _IrrLockTexture($hDesTexture)
```

```
$tSrcPixels = DllStructCreate("uint[" & $iPixelsAmount & "]", $pSrcPixels)
$tDesPixels = DllStructCreate("uint[" & $iPixelsAmount & "]", $pDesPixels)
For $i = 1 To $iPixelsAmount
DllStructSetData($tDesPixels, 1, DllStructGetData($tSrcPixels, 1, $i), $i)
```

```
$pDesPixels += 1
$pSrcPixels += 1
```

Next

```
_IrrUnlockImage($hSrcImage)
_IrrUnlockTexture($hDesTexture)
```

\$tSrcPixels = 0
\$tDesPixels = 0

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(255, 255, 0)
_IrrDraw2DImage($hDesTexture, 0, 0)
_IrrEndScene()
```

WEnd

```
_IrrStop()
Copy to Clipboard
```

IrrUnlockImage

[todo]

#Include <au3Irrlicht2.au3>
_IrrUnlockImage(\$h_Image)

Parameters

\$h_Image		Handle to an irrlicht image object that has been Locked by
	Handle to an irrlicht image object that has been Locked by prior call to _IrrLockImage	

Return Value

Success: True Failure: False and @error > 0

Remarks

None.

Related

<u>IrrLockImage</u>

Example

#include "au3Irrlicht2.au3"

Global \$hSrcImage, \$hDesTexture

```
Global $pSrcPixels, $pDesPixels
Global $tSrcPixels, $tDesPixels
Global $iWidthHeight = 256
Global $iPixelsAmount = $iWidthHeight * $iWidthHeight
```

_IrrStart()

```
$hSrcImage = _IrrGetImage("./media/splatter.tga")
$hDesTexture = _IrrCreateTexture("Desination", $iWidthHeight,
$iWidthHeight, $ECF_A8R8G8B8)
```

```
$pSrcPixels = _IrrLockImage($hSrcImage)
$pDesPixels = _IrrLockTexture($hDesTexture)
```

```
$tSrcPixels = DllStructCreate("uint[" & $iPixelsAmount & "]", $pSrcPixels)
$tDesPixels = DllStructCreate("uint[" & $iPixelsAmount & "]", $pDesPixels)
For $i = 1 To $iPixelsAmount
```

```
$pSrcPixels += 1
```

Next

```
_IrrUnlockImage($hSrcImage)
_IrrUnlockTexture($hDesTexture)
```

```
$tSrcPixels = 0
$tDesPixels = 0
```

```
While _IrrRunning() And Sleep(10)
    _IrrBeginScene(255, 255, 0)
    _IrrDraw2DImage($hDesTexture, 0, 0)
    _IrrEndScene()
WEnd
```

_IrrStop()

IrrCreateRenderTargetTexture

Create a texture that is suitable for the scene manager to use as a surface to which it can render its 3d object.

#Include <au3Irrlicht2.au3>
_IrrCreateRenderTargetTexture(\$i_XSize, \$i_YSize)

Parameters

\$i_XSize	Width of the texture	
\$i_YSize	Height of the texture	

Return Value

Success: Handle to an irrlicht texture object Failure: False and @error 1

Remarks

Each of the dimentions must be of a power of two for example 128x128 or 256x256.

This function is very important when producing texture maps for special effects for example a rendering of a model for a 2D image displayed in the HUD,

the rendering of a model for display on a 3D surface for example a video display of virtual camera, the rendering of the texture for the reflection of a mirror,

the rendering of the environment for use in a water or chrome shader. Most cards, even old cards, will support this very important function.

Related

[todo: functionName, functionName]

IrrMakeNormalMapTexture

Create a normal map from a gray-scale height map texture.

#Include <au3Irrlicht2.au3>
_IrrMakeNormalMapTexture(\$h_Texture, \$f_Amplitude)

Parameters

\$h_Texture	Handle of an device dependend irrlicht texture object
\$f_Amplitude	

Return Value

Success: True Failure: False and @error 1

Remarks

Normal maps are used to add a high level of surface lighting detail to what are normally low resolution models.

They can have a massive effect on the realism of an object, the model you create will have to be created in "tangent" space to support this.

Related

[todo: functionName, functionName]

IrrBlendTextures

Blend the source texture into the destination texture to create a single texture.

#Include <au3Irrlicht2.au3>
_IrrBlendTextures(\$h_TextureDest, \$h_TextureSrc, \$i_Xoffset, \$i_Yoffset,
\$i_Operation)

Parameters

\$h_TextureDest	Handle to the Destination irrlicht texture object.	
\$h_TextureSrc	Handle to the Source irrlicht texture object.	
\$i_Xoffset	X position where the Source texture will be drawn into the Desination texture.	
\$i_Yoffset	Si_Yoffset Y position where the Source texture will be drawn into the Desination texture.	
\$i_Operation	Can be one of the following: \$BLEND_SCREEN ; 0 \$BLEND_ADD ; 1 \$BLEND_SUBTRACT ; 2 \$BLEND_MULTIPLY ; 3 \$BLEND_DIVIDE ; 4	

Return Value

Success: True

Failure: False and set @error, check @extended to see what the error is.@extended 0 then the @error is autoit failed the DIICall@extended 1 Incompatible texture types@extended 2 Unsupported texture format, must be 32bit

Remarks

Textures must be 32 bit format.

Related

<u>_IrrGetTexture</u>, <u>_IrrCreateTexture</u>, <u>_IrrDraw2DImage</u>

Example

#include "au3Irrlicht2.au3"

Global \$hTextureDest, \$hTextureSrc

_IrrStart()

```
$hTextureDest = _IrrGetTexture("./media/Diagonal.bmp")
$hTextureSrc = _IrrGetTexture("./media/cross.bmp")
```

_IrrBlendTextures(\$hTextureDest, \$hTextureSrc, 0, 0, \$BLEND_MULTIPLY)

```
While _IrrRunning() And Sleep(10)
```

```
_IrrBeginScene(50, 50, 50)
_IrrDraw2DImage($hTextureDest, 0, 0)
_IrrEndScene()
```

WEnd

_IrrStop()

```
Copy to Clipboard
```

IrrColorKeyTexture

Copies any parts of the texture that are the same as the specified color into the textures alpha channel.

#Include <au3Irrlicht2.au3>
_IrrColorKeyTexture(\$h_Texture, \$i_Red, \$i_Green, \$i_Blue)

Parameters

\$h_Texture	Handle of the texture object
\$i_Red	Red value from 0 to 255
\$i_Green	Green value from 0 to 255
\$i_Blue	Blue value from 0 to 255

Return Value

Success: True Failure: False

Remarks

This can be used for special effects or to make these regions transparent.

Related

None.

Example

```
#include "au3Irrlicht2.au3"
```

_IrrStart()

```
local $logo = _IrrGetTexture("./media/cross.bmp")
_IrrColorKeyTexture($logo, 255, 255, 255)
```

```
WHILE _IrrRunning()
    _IrrBeginScene(50, 0, 0)
    _IrrDraw2DImageElement( $logo, 0, 0, 0, 0, 128, 64,
$IRR_IGNORE_ALPHA)
    _IrrDraw2DImageElement( $logo, 0, 64, 0, 64, 128, 128,
$IRR_USE_ALPHA)
    _IrrEndScene()
WEND
```

```
_IrrStop()
```

IrrDraw2DImage

Draws the texture to the display at the supplied coordinates.

#Include <au3Irrlicht2.au3>
_IrrDraw2DImage(\$h_Image, \$i_XPos, \$i_YPos)

Parameters

\$h_Image Handle to an irrlicht image object	
\$i_XPos	X position on display from where drawing starts
\$i_YPos	Y position on display from where drawing starts

Return Value

Success: True Failure: False

Remarks

None.

Related

_IrrGetTexture, _IrrDraw2DImageElement

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $logo = _IrrGetTexture("./media/cross.bmp")
```

```
WHILE _IrrRunning()

_IrrBeginScene(50, 0, 0)

_IrrDraw2DImage( $logo, 0, 0 )

_IrrEndScene()

WEND
```

_IrrStop()

```
Copy to Clipboard
```

IrrDraw2DImageElement

Draws given rectangular section from a texture to the display at the supplied coordinates

#Include <au3Irrlicht2.au3>

_IrrDraw2DImageElement(\$h_Texture, \$i_XPos, \$i_YPos, \$i_SourceTopX, \$i_SourceTopY, \$i_SourceBottomX, \$i_SourceBottomY, \$i_UseAlpha)

Parameters

\$h_Image Handle to an irrlicht image object	
\$i_XPos	X position on display from where drawing starts
\$i_YPos	Y position on display from where drawing starts
\$i_SourceTopX	X top position of rectangle in the source texture
\$i_SourceTopY	Y top position of rectangle in the source texture
\$i_SourceBottomX	X bottom position of rectangle in the source texture
\$i_SourceBottomY	Y bottom position of rectangle in the source texture
\$i_UseAlpha	Whether or not to use the alpha channel should be one of the following values: \$IRR_IGNORE_ALPHA \$IRR_USE_ALPHA

Return Value

Success: True Failure: False

Remarks

Draws the texture to the display at the supplied co-ordinates, the image is

copied from the specified rectangle in the source texture, this enables you to put many images onto a single texture.

This function also supports the alpha channel when drawing the image to the display and can draw the image transparently.

Related

```
<u>_IrrGetTexture</u>, <u>_IrrColorKeyTexture</u>, <u>_IrrDraw2DImage</u>,
<u>_IrrDraw2DImageElementStretch</u>
```

Example

```
#include "au3Irrlicht2.au3"
```

_IrrStart()

```
local $logo = _IrrGetTexture("./media/cross.bmp")
_IrrColorKeyTexture($logo, 255, 255, 255)
```

```
WHILE _IrrRunning()
    _IrrBeginScene(50, 0, 0)
    _IrrDraw2DImageElement( $logo, 0, 0, 0, 0, 128, 64,
$IRR_IGNORE_ALPHA)
    _IrrDraw2DImageElement( $logo, 0, 64, 0, 64, 128, 128,
$IRR_USE_ALPHA)
    _IrrEndScene()
WEND
```

_IrrStop()

IrrDraw2DImageElementStretch

Draws specified rectangle from Source texture sizing it to fit the specified Desination rectangle.

#Include <au3Irrlicht2.au3>

_IrrDraw2DImageElementStretch(\$h_Texture, \$i_DestTopX, \$i_DestTopY,

\$i_DestBottomX, \$i_DestBottomY, \$i_SourceTopX, \$i_SourceTopY,

\$i_SourceBottomX, \$i_SourceBottomY, \$i_UseAlpha)

Parameters

\$h_Texture	Handle to an irrlicht image object	
\$i_DestTopX	Top X Destination where the drawing will start.	
\$i_DestTopY	Top Y Destination where the drawing will start.	
\$i_DestBottomX	Bottom X Destination where the drawing will end.	
\$i_DestBottomY	Bottom Y Destination where the drawing will end.	
\$i_SourceTopX	X top position of rectangle in the source texture	
\$i_SourceTopY	Y top position of rectangle in the source texture	
\$i_SourceBottomX	X bottom position of rectangle in the source texture	
\$i_SourceBottomY	Y bottom position of rectangle in the source texture	
\$i_UseAlpha	Whether or not to use the alpha channel should be one of the following values: \$IRR_IGNORE_ALPHA \$IRR_USE_ALPHA	

Return Value

Success: True Failure: False

Remarks

The image is copied from the specified rectangle in the source texture, this enables you to put many images onto a single texture.

If the rectangles are different sizes this function will scale the images appropriately.

This function also supports the alpha channel when drawing the image to the display and can draw the image transparently.

Related

_IrrGetTexture, _IrrColorKeyTexture, _IrrDraw2DImageElement

Example

#include <au3Irrlicht2.au3>

_IrrStart()

Local \$hLogo = _IrrGetTexture("./media/Cross.bmp")

```
; Use White as Alpha color
_IrrColorKeyTexture($hLogo, 255, 255, 255)
```

While _IrrRunning()

_IrrBeginScene(255, 255, 0)

; Draw Original texture just to see what the unchanged texture looks like. _IrrDraw2DImage(\$hLogo, 0, 0)

;Draw texture smaller using Alpha beside the original. _IrrDraw2DImageElementStretch(\$hLogo, 128, 0, 192, 64, 0, 0, 128, 128, \$IRR_USE_ALPHA) ;Draw texture Larger using Alpha beside the previous smaller texture. _IrrDraw2DImageElementStretch(\$hLogo, 192, 0, 448, 256, 0, 0, 128, 128, \$IRR_USE_ALPHA)

;Draw beside Larger texture quarter of source texture to new larger destination not using Alpha.

_IrrDraw2DImageElementStretch(\$hLogo, 448, 0, 800, 352, 0, 0, 64, 64, \$IRR_IGNORE_ALPHA)

_IrrEndScene() WEnd

_IrrStop()

IrrGetFont

Loads a bitmap containing a bitmap font.

```
#Include <au3Irrlicht2.au3>
_IrrGetFont($s_Font)
```

Parameters

\$s_Font	Filename of the bitmap font file
----------	----------------------------------

Return Value

Success: Handle of the irrlicht font texture object Failure: False

Remarks

None

Related

_Irr2DFontDraw

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $bitmapFont = _IrrGetFont ( "./media/fonthaettenschweiler.bmp" )
```

```
WHILE _IrrRunning()
    _IrrBeginScene( 0,0,0 )
    _Irr2DFontDraw ( $BitmapFont, "@! Example Text with 'German
Umlauts': ÄäÖöÜü !@", 120, 80, 250, 96 )
    _IrrEndScene()
WEND
_IrrStop()
```

Irr2DFontDraw

Draws the text into the supplied rectangular area using the supplied font object.

#Include <au3Irrlicht2.au3>
_Irr2DFontDraw(\$h_Font, \$s_Text, \$i_XPos, \$i_YPos, \$i_BottomX,
\$i_BottomY)

Parameters

\$h_Font	Handle of an irrlicht font texture object
\$s_Text	Text string to display
\$i_XPos	X top position of rectangle for the text
\$i_YPos	Y top position of rectangle for the text
\$i_BottomX	X bottom position of rectangle for the text
\$i_BottomY	Y bottom position of rectangle for the text

Return Value

Success: True Failure: False

Remarks

None

Related

_IrrGetFont

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $bitmapFont = _IrrGetFont ( "./media/fonthaettenschweiler.bmp" )
```

```
WHILE _IrrRunning()
    _IrrBeginScene( 0,0,0 )
    _Irr2DFontDraw ( $BitmapFont, "@! Example Text with 'German
Umlauts': ÄäÖöÜü !@", 120, 80, 250, 96 )
    _IrrEndScene()
WEND
```

_IrrStop()

IrrSaveScreenShot

Save a screenshot out to a file.

#Include <au3Irrlicht2.au3>
_IrrSaveScreenShot(\$s_Filename)

Parameters

\$s_Filename file name to save the screenshot as.

Return Value

Success: True Failure: False

Remarks

The image format is defined by the extension applied to the filename.

Irrlicht currently supports: bmp, png, tga, ppm and jpg

Related

_IrrGetScreenShot

Example

#include "au3Irrlicht2.au3"

```
Global $sScreenShot = @MyDocumentsDir & "\IrrScreenShot.jpg"
Global $camera, $mesh, $sceneNode, $texture
```

_IrrStart()

```
; Adding some props to the scene, so we get a screenshot of something.
$camera = _IrrAddCamera(2,2,2, 0,0,0)
$mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
$sceneNode = _IrrAddMeshToScene( $mesh )
$texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
```

```
; We want the props to be drawn before we take a screenshot.
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
```

; Taking a screenshot now the props have been drawn. _IrrSaveScreenShot(\$sScreenShot)

;*If the screenshot was successful then we open it with the users default viewer* If *FileExists*(\$sScreenShot) Then *ShellExecute*(\$sScreenShot)

_IrrStop()

IrrGetScreenShot

Return a pointer to a texture containing a rectangular portion of a screenshot.

#Include <au3Irrlicht2.au3>
_IrrGetScreenShot(\$i_XPos, \$i_YPos, \$i_Width, \$i_Height)

Parameters

\$i_XPos	X position for the screenshot
\$i_YPos	Y position for the screenshot
\$i_Width	Width of the screebshot
\$i_Height	Height of the screebshot

Return Value

Success: Pointer to a texture containing a rectangular portion of a screenshot. Failure: False and @error 1

Remarks

None

Related

_IrrSaveScreenShot, _IrrDraw2DImage, _IrrDraw2DImageElement, _IrrDraw2DImageElementStretch

Example

```
#include "au3Irrlicht2.au3"
```

Global \$hTexture, \$camera, \$mesh, \$sceneNode, \$texture

_IrrStart()

```
; Just adding some props to the scene, so we can get a screenshot of something
$camera = _IrrAddCamera(2,2,2, 0,0,0)
$mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
$sceneNode = _IrrAddMeshToScene( $mesh )
$texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
```

```
WHILE _IrrRunning()
```

_IrrBeginScene(50, 50, 50)

; We want the props to be drawn before we take a screenshot. _IrrDrawScene()

; This will draw our screenshot scaled down to the top left of the window If \$hTexture Then _IrrDraw2DImageElementStretch(\$hTexture, 0, 0, 200, 150, 0, 0, 800, 600, \$IRR_IGNORE_ALPHA)

_IrrEndScene()

; Taking a screenshot just once now the props have been drawn. If Not \$hTexture Then \$hTexture = _IrrGetScreenShot(0, 0, 800, 600)

WEND

_IrrStop()

IrrGetTextureInformation

Get information of a texture. The width, height, pitch and color format is returned in an array.

#Include <au3Irrlicht2.au3>
_IrrGetTextureInformation(\$h_Texture)

Parameters

\$h_Texture Handle to an irrlicht texture object

Return Value

Success: 1D Array with the information \$Array[0] = Width of the texture \$Array[1] = Height of the texture \$Array[2] = Pitch of the texture \$Array[3] = Color Reference of the texture (e.g.: \$ECF_A1R5G5B5, \$ECF_R5G6B5, \$ECF_R8G8B8, \$ECF_A8R8G8B8) Failure: Empty Array and Sets @error to 1

Remarks

This function cannot be used for image objects. For this, use _IrrGetImageInformation instead.

Related

_IrrGetTexture, _IrrGetImageInformation

Example

#include <au3Irrlicht2.au3>
_IrrStart()
Local \$hLogo = _IrrGetTexture("./media/Cross.bmp")
Local \$aInfo = _IrrGetTextureInformation(\$hLogo)
If Not @error Then MsgBox(64, "Texture Information", "Width: " & \$aInfo[0]
& @LF & ______
"Height: " & \$aInfo[1] & @LF & "Pitch: " & \$aInfo[2] &
@LF & ______
"Color Format: " & \$aInfo[3])
_IrrStop()
Copy to Clipboard

IrrGetImageInformation

Get information of an image. The width, height, pitch and color format is returned in an array.

#Include <au3Irrlicht2.au3>
_IrrGetImageInformation(\$h_Image)

Parameters

\$h_Texture Handle to an irrlicht image object

Return Value

Success: 1D Array with the information \$Array[0] = Width of the image \$Array[1] = Height of the image \$Array[2] = Pitch of the image \$Array[3] = Color Reference of the image (e.g.: \$ECF_A1R5G5B5, \$ECF_R5G6B5, \$ECF_R8G8B8, \$ECF_A8R8G8B8) Failure: Empty Array and Sets @error to 1

Remarks

This function cannot be used for texture objects. For this, use _IrrGetTextureInformation instead.

Related

_IrrGetImage, _IrrGetTextureInformation

Example

#include <au3Irrlicht2.au3>
_IrrStart()
Local \$hLogo = _IrrGetImage("./media/Cross.bmp")
Local \$aInfo = _IrrGetImageInformation(\$hLogo)
If Not @error Then MsgBox(64, "Image Information", "Width: " & \$aInfo[0]
& @LF & ______
"Height: " & \$aInfo[1] & @LF & "Pitch: " & \$aInfo[2] &
@LF & ______
"Color Format: " & \$aInfo[3])
_IrrStop()
Copy to Clipboard

IrrSetNodeAnimationRange

Sets the range of animation that is to be played in the node.

#Include <au3Irrlicht2.au3>
_IrrSetNodeAnimationRange(\$h_Node, \$i_Start, \$i_End)

Parameters

\$h_Node	Handle to a character scene node.	
\$i_Start	Start frame	
\$i_End	End frame	

Return Value

Success: True Failure: False

Remarks

An anaimation sequences might run from 0 to 200 frames and a sequence where your character is running might only occupy a portion of this.

Related

<u>IrrGetMesh</u>, <u>IrrAddMeshToScene</u>, <u>IrrGetMeshFrameCount</u>, <u>IrrGetNodeAnimationFrame</u>, <u>IrrPlayNodeMD2Animation</u>, <u>IrrSetNodeAnimationSpeed</u>

Example

```
#include "au3Irrlicht2.au3"
```

Global \$hMD2Mesh, \$hMeshTexture, \$hSceneNode

_IrrStart()

```
$hMD2Mesh = _IrrGetMesh("./media/zumlin.md2")
$hMeshTexture = _IrrGetTexture("./media/zumlin.pcx")
$hSceneNode = _IrrAddMeshToScene($hMD2Mesh)
_IrrSetNodeMaterialTexture($hSceneNode, $hMeshTexture, 0)
_IrrSetNodeMaterialFlag($hSceneNode, $IRR_EMF_LIGHTING,
$IRR_OFF)
```

```
_IrrSetNodeAnimationRange($hSceneNode, 200, 230)
```

```
_IrrAddCamera(50, 0, 0, 0, 0, 0)
```

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(240, 255, 255)
_IrrDrawScene()
_IrrEndScene()
WEnd
```

```
_IrrStop()
Copy to Clipboard
```

IrrPlayNodeMD2Animation

Selects the animation sequence of MD2 to be played.

#Include <au3Irrlicht2.au3>
_IrrPlayNodeMD2Animation(\$h_Node, \$i_Animation)

Parameters

\$h_Node Handle to a character scene node.	
\$n_Node Handle to a character scene node. sequence should be one of the following \$IRR_EMAT_STAND \$IRR_EMAT_RUN \$IRR_EMAT_RUN \$IRR_EMAT_ATTACK \$IRR_EMAT_PAIN_A \$IRR_EMAT_PAIN_A \$IRR_EMAT_PAIN_C \$IRR_EMAT_PAIN_C \$IRR_EMAT_TOPAIN_C \$IRR_EMAT_SALUTE \$IRR_EMAT_SALUTE \$IRR_EMAT_SALUTE \$IRR_EMAT_CROUCH_STAND \$IRR_EMAT_CROUCH_STAND \$IRR_EMAT_CROUCH_ATTACK \$IRR_EMAT_CROUCH_MALK \$IRR_EMAT_CROUCH_ATTACK \$IRR_EMAT_CROUCH_PAIN \$IRR_EMAT_CROUCH_DEATH \$IRR_EMAT_CROUCH_DEATH \$IRR_EMAT_CROUCH_DEATH \$IRR_EMAT_DEATH_FALLBACK \$IRR_EMAT_DEATH_FALLBACK \$IRR_EMAT_DEATH_FALLBACKSLC \$IRR_EMAT_DEATH_FALLBACKSLC	D

Return Value

Success: True Failure: False

Remarks

MD2 format models have specific animation sequences contained within them that can be played back with a simple call.

Related

_IrrGetMesh, _IrrAddMeshToScene, _IrrGetMeshFrameCount, _IrrGetNodeAnimationFrame, _IrrSetNodeAnimationRange, _IrrSetNodeAnimationSpeed

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $MD2Mesh = _IrrGetMesh( "./media/zumlin.md2" )
local $MeshTexture = _IrrGetTexture( "./media/zumlin.pcx" )
local $SceneNode = _IrrAddMeshToScene( $MD2Mesh )
_IrrSetNodeMaterialTexture( $SceneNode, $MeshTexture, 0 )
_IrrSetNodeMaterialFlag( $SceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
```

_IrrPlayNodeMD2Animation(\$SceneNode, \$IRR_EMAT_SALUTE)

```
local $camera = _IrrAddCamera( 50,0,0, 0,0,0 )
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
```

_IrrDrawScene() _IrrEndScene() WEND

IrrSetNodeAnimationSpeed

Change the speed at which an animation is played for a node

#Include <au3Irrlicht2.au3>
_IrrSetNodeAnimationSpeed(\$h_Node, \$f_Speed)

Parameters

\$h_Node	Handle to a character scene node.
\$f_Speed	How many frames per second.

Return Value

Success: True Failure: False

Remarks

You could use this to make a character run slowly or quickly and still keep its feet on the ground.

Related

<u>_IrrGetMesh</u>, <u>_IrrAddMeshToScene</u>, <u>_IrrGetMeshFrameCount</u>, <u>_IrrGetNodeAnimationFrame</u>, <u>_IrrSetNodeAnimationRange</u>

Example

#include "au3Irrlicht2.au3"

Global \$hMD2Mesh, \$hMeshTexture, \$hSceneNode

_IrrStart()

```
$hMD2Mesh = _IrrGetMesh("./media/zumlin.md2")
$hMeshTexture = _IrrGetTexture("./media/zumlin.pcx")
$hSceneNode = _IrrAddMeshToScene($hMD2Mesh)
_IrrSetNodeMaterialTexture($hSceneNode, $hMeshTexture, 0)
_IrrSetNodeMaterialFlag($hSceneNode, $IRR_EMF_LIGHTING,
$IRR_OFF)
```

_IrrSetNodeAnimationSpeed(\$hSceneNode, 75)

_IrrAddCamera(50, 0, 0, 0, 0, 0)

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(240, 255, 255)
_IrrDrawScene()
_IrrEndScene()
WEnd
```

_IrrStop()

IrrGetNodeAnimationFrame

Get the frame number that is currently being played by the node.

#Include <au3Irrlicht2.au3>
_IrrGetNodeAnimationFrame(\$h_Node)

Parameters

\$h_Node Handle to a character scene node.

Return Value

Success: Current frame number being played. Failure: False and @error 1

Remarks

None.

Related

<u>_IrrGetMesh</u>, <u>_IrrAddMeshToScene</u>, <u>_IrrGetMeshFrameCount</u>, <u>_IrrSetNodeAnimationSpeed</u>, <u>_IrrSetNodeAnimationRange</u>

Example

#include "au3Irrlicht2.au3"

Global \$hMD2Mesh, \$hMeshTexture, \$hSceneNode

_IrrStart()

```
$hMD2Mesh = _IrrGetMesh("./media/zumlin.md2")
$hMeshTexture = _IrrGetTexture("./media/zumlin.pcx")
$hSceneNode = _IrrAddMeshToScene($hMD2Mesh)
_IrrSetNodeMaterialTexture($hSceneNode, $hMeshTexture, 0)
_IrrSetNodeMaterialFlag($hSceneNode, $IRR_EMF_LIGHTING,
$IRR_OFF)
_IrrAddCamera(50, 0, 0, 0, 0, 0)
```

While _IrrRunning() And Sleep(10)
 _IrrBeginScene(240, 255, 255)
 _IrrSetWindowCaption("_IrrGetNodeAnimationFrame - Playing Frame: "
& _IrrGetNodeAnimationFrame(\$hSceneNode))
 _IrrDrawScene()
 _IrrEndScene()
WEnd
_IrrStop()
Copy to Clipboard

IrrSetNodeAnimationFrame

Set the current frame number being played in the animation.

#Include <au3Irrlicht2.au3>
_IrrSetNodeAnimationFrame(\$h_Node, \$f_Frame)

Parameters

\$h_Node	Handle to a character scene node.	
\$f_Frame	Sf_Frame Frame number to play	

Return Value

Success: True Failure: False

Remarks

None.

Related

_IrrGetMesh, _IrrAddMeshToScene, _IrrGetMeshFrameCount

Example

#include "au3Irrlicht2.au3"

Global \$hMD2Mesh, \$hMeshTexture, \$hSceneNode

_IrrStart()

```
$hMD2Mesh = _IrrGetMesh("./media/zumlin.md2")
$hMeshTexture = _IrrGetTexture("./media/zumlin.pcx")
$hSceneNode = _IrrAddMeshToScene($hMD2Mesh)
_IrrSetNodeMaterialTexture($hSceneNode, $hMeshTexture, 0)
_IrrSetNodeMaterialFlag($hSceneNode, $IRR_EMF_LIGHTING,
$IRR_OFF)
```

```
_IrrSetNodeAnimationFrame($hSceneNode, 600)
```

_IrrAddCamera(50, 0, 0, 0, 0, 0)

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(240, 255, 255)
_IrrDrawScene()
_IrrEndScene()
```

WEnd

_IrrStop()

Copy to Clipboard

IrrSetTransitionTime

Sets the transition time across which two poses of an animated mesh are blended.

#Include <au3Irrlicht2.au3>
_IrrSetTransitionTime(\$h_Node, \$f_Speed)

Parameters

\$h_Node	Handle to a character scene node.	
\$f_Speed	f_Speed Speed of the transition	

Return Value

Success: True Failure: False

Remarks

For example a character in a sitting pose can be switched into a lying down pose by blending the two frames,

this will provide a more convincing smooth transition instead of a snap change in position.

_IrrAnimateJoints must be called before IrrDrawScene if blending is used.

Related

_IrrAnimateJoints, _IrrSetJointMode

IrrAnimateJoints

Animates the mesh based on the position of the joints.

#Include <au3Irrlicht2.au3>
_IrrAnimateJoints(\$h_Node)

Parameters

\$h_Node Handle to a character scene node.

Return Value

Success: True Failure: False

Remarks

This should be used at the end of any manual joint operations including blending and joints animated using IRR_JOINT_MODE_CONTROL and _IrrSetNodeRotation on a bone node.

Related

_IrrSetNodeRotation, _IrrSetJointMode

IrrSetJointMode

Sets the animation mode of joints in a node.

#Include <au3Irrlicht2.au3>
_IrrSetJointMode(\$h_Node, \$i_Mode)

Parameters

\$h_Node	Handle to a character scene node.	
\$i_Mode	can be one of the following; \$IRR_JOINT_MODE_NONE - no animation of the model based on bones. \$IRR_JOINT_MODE_READ - automatic animation based upon the animation defined with calls like _IrrSetNodeAnimationRange. IRR_JOINT_MODE_CONTROL - allow the position of the bones to be set through code.	

Return Value

Success: True Failure: False

Remarks

When using the control mode _IrrAnimateJoints must be called before IrrDrawScene.

Related

_IrrSetNodeAnimationRange, _IrrAnimateJoints, _IrrSetTransitionTime

IrrAddCollisionAnimator

Animator applying collision detection and gravity to its parent node.

#Include <au3Irrlicht2.au3>
_IrrAddCollisionAnimator(\$h_IrrSelector, \$h_Node, \$f_RadiusX,
\$f_RadiusY, \$f_RadiusZ, \$f_GravityX, \$f_GravityY, \$f_GravityZ,
\$f_OffsetX, \$f_OffsetY, \$f_OffsetZ)

Parameters

\$h_IrrSelector	Handle of a selecor object as created with _IrrGetCollision[] The selector represents a selection of triangles in the scene, this is usually all of the triangles in a map for instance.	
\$h_Node	Handle of a scene node to be collided against the selector.	
\$f_RadiusX, \$f_RadiusY, \$f_RadiusZ	Define an ellipsoid that defines the area of collision this eliptical shape allows the collision detection to slide the object up steps and even ladders. If you make it too big you might be too large to get through a doorway but if you make it too small you may not be able to climb steps. You should play with these values and find the best ones for your scene.	
\$f_GravityX, \$f_GravityY, \$f_GravityZ	Specify the force that is applied to the node for each axis. For example 0.0,-9.8,0.0 defines a typical downward force. Other values could be used to simulate e.g. wind effects.	
\$f_OffsetX, \$f_OffsetY, \$f_OffsetZ	Offset the node by a specific distance from the center of the collision. As the center of the object and the size of your collision ellipsoid vary you can use this to adjust the position of	

Return Value

Success: Handle of the created animator. Failure: False

Remarks

The collision detection will stop the object penetrating through a surface in the objects it is colliding against and will also press it against the surface using gravity.

Related

_IrrRemoveAnimator, _IrrGetCollisionGroupFromMesh,

_IrrGetCollisionGroupFromComplexMesh,

_IrrGetCollisionGroupFromBox, _IrrGetCollisionGroupFromTerrain,

IrrCreateCombinedCollisionGroup

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
_IrrAddZipFile( "./media/map-20kdm2.pk3", $IRR_IGNORE_CASE,
$IRR_IGNORE_PATHS )
$meshBSP = _IrrGetMesh( "20kdm2.bsp" )
local $nodeBSP = _IrrAddMeshToSceneAsOcttree( $meshBSP )
```

```
local $nodeCamera = _IrrAddFPSCamera()
_IrrSetNodePosition( $nodeCamera, 1750, 149, 1369 )
```

IrrAddDeleteAnimator

Animator deleting its parent node after specified time (ms).

#Include <au3Irrlicht2.au3>
_IrrAddDeleteAnimator(\$h_Node, \$i_Time)

Parameters

\$h_Node	Handle of a scene Node.	
\$i_Time Living time of the animator in milliseconds.		

Return Value

Success: Handle of the created animator. Failure: False

Remarks

You could use this animator to delete a falling rock for example, all you would need to do is attach the delete animator, a movement animator and then forget about it.

Related

IrrRemoveAnimator

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCamera = _IrrAddCamera(10,10,10, 0,0,0 )
local $nodeTest = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
_IrrAddDeleteAnimator($nodeTest, 3000)
WHILE _IrrRunning()
_ L_D__ri_ for ___ (0, 0, 0)
```

```
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
```

```
_IrrStop()
```

Copy to Clipboard

IrrAddFlyCircleAnimator

Animator moving its parent node in a circular path.

#Include <au3Irrlicht2.au3>
_IrrAddFlyCircleAnimator(\$h_Node, \$f_CenterX, \$f_CenterY, \$f_CenterZ,
\$f_Radius, \$f_Speed)

Parameters

\$h_Node	Handle of a scene Node.	
\$f_CenterX, \$f_CenterY, \$f_CenterZ	Define center of the circular path.	
\$f_Radius	Defines the radius of the circular path.	
\$f_Speed	Defines how far the node is moved each frame.	

Return Value

Success: Handle of the created animator. Failure: False

Remarks

None

Related

<u>IrrRemoveAnimator</u>

Example

```
#include "au3Irrlicht2.au3"
```

_IrrStart()

```
local $nodeCamera = _IrrAddCamera(10,10,10, 0,0,0 )
local $nodeTest = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
```

_IrrAddFlyCircleAnimator(\$nodeTest, 0, 0, 0, 5, 0.005)

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
```

WEND

_IrrStop()

Copy to Clipboard

IrrAddFlyStraightAnimator

Animator moving its parent node in a straight line from a start to an end point.

#Include <au3Irrlicht2.au3>

_IrrAddFlyStraightAnimator(\$h_Node, \$f_StartX, \$f_StartY, \$f_StartZ, \$f_EndX, \$f_EndY, \$f_EndZ, \$i_Time, \$i_DoLoop)

Parameters

\$h_Node	Handle of a scene Node.
\$f_StartX, \$f_StartY, \$f_StartZ	Specify the start point of the path.
\$f_EndX, \$f_EndY, \$f_EndZ	Specify the end point of the path.
\$i_Time	Number of milliseconds the animator will take to move the node from start to end point.
\$i_DoLoop	Determines single or continously movement type: \$IRR_ONE_SHOT - For a single animation and then stop. \$IRR_LOOP - To continuously repeat the animation.

Return Value

Success: Handle of the created animator. Failure: False

Remarks

None

Related

_IrrRemoveAnimator

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $nodeCamera = _IrrAddCamera(10,10,10,0,0,0)
local $nodeTest = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
_IrrAddFlyStraightAnimator($nodeTest, -25, -10, -10, 25, 10, 2, 5000,
$IRR_LOOP)
WHILE _IrrRunning()
  _IrrBeginScene(0, 0, 0)
  _IrrDrawScene()
  _IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrAddRotationAnimator

Animator spinning its parent node around each axis.

#Include <au3Irrlicht2.au3>
_IrrAddRotationAnimator(\$h_Node, \$f_RotX, \$f_RotY, \$f_RotZ)

Parameters

\$h_Node	Handle of a scene Node.
\$f_RotX, \$f_RotY, \$f_RotZ	Specify number of radians the object is spun around each axis.

Return Value

Success: Handle of the created animator. Failure: False

Remarks

None

Related

<u>IrrRemoveAnimator</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCamera = _IrrAddCamera(10,10,10,0,0,0)
local $nodeTest = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
_IrrAddRotationAnimator($nodeTest, 0.1, 0.2, 0.3)
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrAddSplineAnimator

Animator moving its parent node along a spline curve.

#Include <au3Irrlicht2.au3>
_IrrAddSplineAnimator(\$h_Node, \$tVectorStruct, \$i_Start, \$f_Speed,
\$f_Tightness)

Parameters

\$h_Node	Handle of a scene Node.	
\$tVectorStruct	Struct array with 3D-points defining the spline curve. UseCreateVectStruct andSetVectStruct to build the required struct.	
\$i_Start	Time in milliseconds that must pass before the animation starts.	
\$f_Speed	Defines the rate the node moves along the spline curve.	
\$f_Tightness	Specifies how tightly the curve is tied to the points. Value between 0 (angular) and 1 (very loose).	

Return Value

Success: Handle of the created animator. Failure: False and sets @error:

- 1 error from .dll call
- 2 \$tVectorStruct is not a dllstruct

Remarks

This is one of the more difficult to set up of the animators but is very natural looking and powerful.

A spline is a curved line that passes through or close to a list of co-

ordinates, creating a smooth flight.

This animator needs a list of coordinates stored in a struct array for the X, Y and Z locations of all the points.

A good way to get coordinates for this struct is to load in the camera position example and move your camera to a point and write down its coordinates.

Related

<u>CreateVectStruct</u>, <u>SetVectStruct</u>, <u>IrrRemoveAnimator</u>

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $nodeCamera = _IrrAddCamera(150,50,0, 0,75,0)
local $nodeTest = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
local $tVectors = __CreateVectStruct(4)
__SetVectStruct($tVectors, 0, -100, 50, 0)
__SetVectStruct($tVectors, 1, 0, 100, -100)
__SetVectStruct($tVectors, 3, 0, 100, 100)
```

_IrrAddSplineAnimator(\$nodeTest, \$tVectors, 0, 0.5, 1)

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
```

WEND

_IrrStop()

Copy to Clipboard

IrrAddFadeAnimator

This animator deletes the node it is attached to after the specified number of milliseconds.

```
#Include <au3Irrlicht2.au3>
_IrrAddFadeAnimator($h_Node, $i_DeleteAfterMiliseconds, $f_Scale =
1.0)
```

Parameters

· _	Handle of a scene Node.
\$i_DeleteAfterMiliseconds	Number of milliseconds before deleting the node.
1	Scaled amount while fading.

Return Value

Success: Handle of the created animator. Failure: False

Remarks

During the time while it is waiting to delete it the node is slowly faded to invisibility and is also scaled by the specified amount.

You could use this animator to fade and delete an object from a scene that was no longer required like a used medical pack,

all you would need to do is attach the fade animator and forget about it.

Related

_IrrAddDeleteAnimator, _IrrRemoveAnimator

Example

```
#include "au3Irrlicht2.au3"
Global $hMD2Mesh, $hMeshTexture, $hSceneNode
_IrrStart()
$hMD2Mesh = _IrrGetMesh("./media/zumlin.md2")
$hMeshTexture = _IrrGetTexture("./media/zumlin.pcx")
$hSceneNode = IrrAddMeshToScene($hMD2Mesh)
_IrrSetNodeMaterialTexture($hSceneNode, $hMeshTexture, 0)
_IrrSetNodeMaterialFlag($hSceneNode, $IRR_EMF_LIGHTING, $IRR_ON)
_IrrSetAmbientLight( 1,1,1 )
IrrAddFadeAnimator($hSceneNode, 3000, -1)
_IrrAddCamera(50, 0, 0, 0, 0, 0)
While _IrrRunning() And Sleep(10)
  _IrrBeginScene(240, 255, 255)
  _IrrDrawScene()
  _IrrEndScene()
WEnd
_IrrStop()
```

Copy to Clipboard

IrrRemoveAnimator

Removes an applied animator from a node.

#Include <au3Irrlicht2.au3>
_IrrRemoveAnimator(\$h_Node, \$h_Animator)

Parameters

\$h_Node	Handle of a scene Node.
\$h_Animator	Handle of the animator to be removed.

Return Value

Success: True Failure: False

Remarks

After the function call a played animation is stopped or the animator cleaned up so you can apply a new one.

Related

<u>_IrrAddCollisionAnimator</u>, <u>_IrrAddDeleteAnimator</u>, <u>_IrrAddFlyCircleAnimator</u>, <u>_IrrAddFlyStraightAnimator</u>, <u>_IrrAddRotationAnimator</u>, <u>_IrrAddSplineAnimator</u>, <u>_IrrAddFadeAnimator</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCamera = _IrrAddCamera(10,10,10, 0,0,0 )
local $nodeTest = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
local $animator = _IrrAddFlyCircleAnimator($nodeTest, 0, 0, 0, 5, 0.005)
```

local \$timer = TimerInit()

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
```

if TimerDiff(\$timer) > 3000 then _IrrRemoveAnimator(\$nodeTest,
\$animator)
WEND

_IrrStop()

```
Copy to Clipboard
```

IrrAddFPSCamera

Adds a 'first person shooter' style camera with mouse and keyboard control into the scene.

#Include <au3Irrlicht2.au3>
_IrrAddFPSCamera(\$h_ParentNode = 0, \$f_RotateSpeed = 100.0,
\$f_MoveSpeed = 0.5, \$i_ID = -1, \$h_KeyMapArray = 0, \$i_KeyMapSize =
0, \$i_NoVerticalMovement = 0, \$f_JumpSpeed = 0.0)

Parameters

\$h_ParentNode	[optional] Parent scene node of the camera. Can be null.
\$f_RotateSpeed	[optional] Speed in degress with which the camera is rotated. This can be done only with the mouse.
\$f_MoveSpeed	[optional] Speed in units per millisecond with which the camera is moved. Movement is done with the cursor keys.
\$i_ID	[optional] id of the camera. This id can be used to identify the camera.
\$h_KeyMapArray	[optional] Adress of a key map as created withCreatePtrKeyMapArray, specifying what keys should be used to move the camera. If this is null, the default keymap is used. You can define actions more then one time in the array, to bind multiple keys to the same action.
\$i_KeyMapSize	[optional] Amount of items in the keymap array.

	[optional] Setting this to true makes the camera only move within a horizontal plane, and disables vertical movement as known from most ego shooters. Default is 'false', with which it is possible to fly around in space, if no gravity is there.
\$f_JumpSpeed	[optional] Speed with which the camera is moved when jumping.

Return Value

Success: Handle of the camera object Failure: False

Remarks

Adds a camera scene node with an animator which provides mouse and keyboard control appropriate for first person shooters (FPS).

If however you capture events when starting irrlicht this will become a normal camera that can only be moved by code.

This FPS camera is intended to provide a demonstration of a camera that behaves like a typical First Person Shooter.

It is useful for simple demos and prototyping but is not intended to provide a full solution for a production quality game.

It binds the camera scene node rotation to the look-at target.

Related

<u>_CreatePtrKeyMapArray, _IrrAddCamera, _IrrAddMayaCamera</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $camera = _IrrAddFPSCamera()
```

```
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $sceneNode = _IrrAddMeshToScene( $mesh )
_IrrSetNodePosition($sceneNode, 0, 0, 5 )
```

```
local $texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
```

```
_IrrSetWindowCaption("Move with mouse + cursor keys - quit with ALT-
F4!")
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrAddCamera

Adds a camera into the scene.

```
#Include <au3Irrlicht2.au3>
_IrrAddCamera($f_CamX, $f_CamY, $f_CamZ, $f_TargetX, $f_TargetY,
$f_TargetZ)
```

Parameters

\$f_CamX	X value for view point of the camera.
\$f_CamY	Y value for view point of the camera.
\$f_CamZ	Z value for view point of the camera.
\$f_TargetX	X value for target of camera.
\$f_TargetY	Y value for target of camera.
\$f_TargetZ	Z value for target of camera.

Return Value

Success: Handle of the camera object Failure: False

Remarks

The camera will be used to define the view point and target point and other attributes of the view into the 3D scene. Animators and other node functions can be applied to this node.

Related

_IrrAddFPSCamera, _IrrAddMayaCamera

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $camera = _IrrAddCamera(2,2,2, 0,0,0 )
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $sceneNode = _IrrAddMeshToScene( $mesh )
local $texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
WHILE _IrrRunning()
  _IrrBeginScene(50, 50, 50)
  _IrrDrawScene()
  _IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrAddMayaCamera

Adds a Maya style camera to into the scene.

```
#Include <au3Irrlicht2.au3>
_IrrAddMayaCamera($h_Node = $IRR_NO_OBJECT, $f_Rotate = 100.0,
$f_Zoom = 100.0, $f_Move = 100.0)
```

Parameters

\$h_Node	Handle parent irr node, if no parent is needed then use \$IRR_NO_OBJECT.
\$f_Rotate	Speed at which the camera revolves.
\$f_Zoom	Speed at which the camera zooms in and out.
\$f_Move	Speed at which the camera moves.

Return Value

Success: Handle of the camera object Failure: False

Remarks

The user can click with the left, middle and right mouse buttons to move, zoom and rotate the camera.

Related

_IrrAddCamera, _IrrAddFPSCamera

Example

```
#include "au3Irrlicht2.au3"
```

Global \$hMD2Mesh, \$hMeshTexture, \$hSceneNode, \$hMayaCamera

_IrrStart()

```
$hMD2Mesh = _IrrGetMesh("./media/zumlin.md2")
$hMeshTexture = _IrrGetTexture("./media/zumlin.pcx")
$hSceneNode = _IrrAddMeshToScene($hMD2Mesh)
_IrrSetNodeMaterialTexture($hSceneNode, $hMeshTexture, 0)
_IrrSetNodeMaterialFlag($hSceneNode, $IRR_EMF_LIGHTING,
$IRR_OFF)
```

```
$hMayaCamera = _IrrAddMayaCamera($hSceneNode)
_IrrSetCameraTarget($hMayaCamera, 0, 0, 0)
```

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(240, 255, 255)
_IrrDrawScene()
_IrrEndScene()
WEnd
```

_IrrStop()
Copy to Clipboard

IrrSetCameraTarget

Sets the point in space that the camera is looking at.

#Include <au3Irrlicht2.au3>
_IrrSetCameraTarget(\$h_Camera, \$f_CamX, \$f_CamY, \$f_CamZ)

Parameters

\$h_Camera	Handle of a camera object
\$f_CamX, \$f_CamY, \$f_CamZ	Position in the scene to target with the camera.

Return Value

Success: True Failure: False

Remarks

The camera view point can be moved by simply using the _IrrSetNodePosition function but this operation will change the point that the camera is pointing at.

Related

<u>IrrGetCameraTarget</u>, <u>IrrSetCameraUpAtRightAngle</u>, <u>IrrSetCameraUpDirection</u>, <u>IrrGetCameraOrientation</u>, <u>IrrGetCameraUpDirection</u>

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
; set up a sphere into the scene:
local $nodeSphere = _IrrAddSphereSceneNode(5)
_IrrSetNodeMaterialTexture( $nodeSphere,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeSphere, $IRR_EMF_LIGHTING, $IRR_OFF
)
_IrrSetNodePosition($nodeSphere, 10, 5, 10)
local $nodeCamera = _IrrAddFPSCamera()
; target the sphere with the camera:
_IrrSetCameraTarget($nodeCamera, 10, 5, 10)
WHILE _IrrRunning()
  _IrrBeginScene(0, 0, 25)
  _IrrDrawScene()
  _IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrGetCameraTarget

Returns array with coordinates of point in space that the camera is looking at.

#Include <au3Irrlicht2.au3>
_IrrGetCameraTarget(\$h_Camera)

Parameters

\$h_Camera Handle of a camera object

Return Value

success: Array with three elements for X, Y, Z. failure: False

Remarks

None.

Related

```
_IrrSetCameraTarget, _IrrSetCameraUpAtRightAngle,
_IrrSetCameraUpDirection, _IrrGetCameraOrientation,
_IrrGetCameraUpDirection
```

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
_IrrAddTestSceneNode()
local $nodeCamera = _IrrAddFPSCamera()
_IrrSetNodePosition($nodeCamera, 0, 0, -50)
```

local \$aTarget

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 50)
```

```
$aTarget = _IrrGetCameraTarget($nodeCamera)
_IrrSetWindowCaption('Targeting: X: ' & int($aTarget[0]) & ' / Y: ' &
int($aTarget[1]) & ' / Z: ' & int($aTarget[2]) )
```

```
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrGetCameraOrientation

Gets the camera orientation (forward, upward and sideways vectors of the camera)

#Include <au3Irrlicht2.au3>
_IrrGetCameraOrientation(\$h_Camera, ByRef \$a_Vector1, ByRef
\$a_Vector2, ByRef \$a_Vector3)

Parameters

\$h_Camera	Handle of a camera object
\$a_Vector1, \$a_Vector2, \$a_Vector3	Any variables to populate with the camera orientation vectors, must not explicitly be arrays.

Return Value

Success: True Failure: False

Remarks

Returned vectors will be different lengths depending on how much the camera is rotated The described camera direction is useful after the camera has been revolved.

Related

<u>_IrrGetCameraTarget</u>, <u>_IrrSetCameraTarget</u>, <u>_IrrSetCameraUpAtRightAngle</u>, <u>_IrrSetCameraUpDirection</u>, <u>_IrrGetCameraUpDirection</u>

Example

#include "au3Irrlicht2.au3"

; better example welcome :)

_IrrStart()

_IrrAddTestSceneNode()
local \$nodeCamera = _IrrAddFPSCamera()
_IrrSetNodePosition(\$nodeCamera, 0, 0, -50)

local \$aVectForward, \$aVectUpward, \$aVectSide

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 50)
```

```
_IrrGetCameraOrientation($nodeCamera, $aVectForward, $aVectUpward,
$aVectSide)
_IrrSetWindowCaption('Upward vector: ' & _
$aVectUpward[0] & ' / Y: ' & $aVectUpward[1] & ' / Z: ' &
$aVectUpward[2] )
```

```
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrRevolveCamera

Revolve the camera using quaternion calculations.

#Include <au3Irrlicht2.au3>
_IrrRevolveCamera(\$h_Camera, \$f_Yaw, \$f_Pitch, \$f_Roll, \$f_Drive,
\$f_Strafe, \$f_Elevate)

Parameters

\$h_Camera	Handle of a camera node.
\$f_Yaw	Affects turning left and right.
\$f_Pitch	Affects tilting up and down.
\$f_Roll	Affects rolling left and right.
\$f_Drive	Affects moving forwards and backward.
\$f_Strafe	Affects moving left and right.
\$f_Elevate	Affects moving up and down.

Return Value

Success: True Failure: False

Remarks

This will help avoid gimbal lock associated with normal Rotations and is ideal for spacecraft and aircraft.

Related

IrrSetCameraUpAtRightAngle

Set the camera up at a right angle to the camera vector.

#Include <au3Irrlicht2.au3>
_IrrSetCameraUpAtRightAngle(\$h_Camera)

Parameters

\$h_Camera Handle of a camera node.

Return Value

Success: True Failure: False

Remarks

[todo]

Related

IrrSetCameraOrthagonal

Set the projection of the camera to an orthagonal view, where there is no sense of perspective.

#Include <au3Irrlicht2.au3>
_IrrSetCameraOrthagonal(\$h_Camera, \$f_DistanceX, \$f_DistanceY,
\$f_DistanceZ)

Parameters

\$h_Camera	Handle of a camera node.
\$f_DistanceX	X Distance
\$f_DistanceY	Y Distance
\$f_DistanceZ	Z Distance

Return Value

Success: True Failure: False

Remarks

The distance to the target adjusts the width and height of the camera view, essentially the smaller it is the larger the object will appear.

Related

IrrSetCameraClipDistance

Defines far and near distances for camera clipping

#Include <au3Irrlicht2.au3>
_IrrSetCameraClipDistance(\$h_Camera, \$f_Distance, \$f_NearDistance =
1.0)

Parameters

\$h_Camera	Handle of a camera node
\$f_Distance	Defines the far distance for clipping
\$f_NearDistance	[optional] Defines the near distance for clipping (towards the camera)

Return Value

Success: True Failure: False

Remarks

The clipping distances of a camera are the distances beyond and before which no triangles are rendered. Default clipping is before 1.0 and behind 2000.0.

This speeds the scene up by not showing geometry that is beyond or before the defined distances and increases rendering performance without requiring you to manage adding and deleting the objects from the view.

To make the far distance clipping less abrupt you can use it in combination with _IrrSetFog.

Related

IrrAddCamera, IrrAddFPSCamera, IrrSetFog

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $nodeCube = _IrrAddCubeSceneNode(12)
_IrrSetNodeMaterialTexture( $nodeCube,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeCube, $IRR_EMF_LIGHTING, $IRR_OFF )
local $nodeCamera = _IrrAddCamera(10, 10, 10, 0, 0, 0)
_IrrSetCameraClipDistance($nodeCamera, 15)
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrSetActiveCamera

When you have several camera objects in the scene you can use this call to define which of them is to be used to look through when drawing the scene.

#Include <au3Irrlicht2.au3>
_IrrSetActiveCamera(\$h_Camera)

Parameters

\$h_Camera Handle of a camera node.

Return Value

Success: True Failure: False

Remarks

None.

Related

IrrSetCameraFOV

Sets the field of vision of the camera.

#Include <au3Irrlicht2.au3>
_IrrSetCameraFOV(\$h_Camera, \$f_FOV)

Parameters

\$h_Camera	Handle of a camera node.
\$f_FOV	The value is in radians and has a default value of PI / 2.5

Return Value

Success: True Failure: False

Remarks

A wide field of vision will give a distorted perspective, if the angle is too narrow the display will feel restricted.

Related

IrrSetCameraAspectRatio

Sets the aspect ratio of the camera in the same way you think of standard screens and widescreens.

#Include <au3Irrlicht2.au3>
_IrrSetCameraAspectRatio(\$h_Camera, \$f_AspectRatio)

Parameters

\$h_Camera	Handle of a camera node.
\$f_AspectRatio	Aspect ratio as a float value.

Return Value

Success: True Failure: False

Remarks

A widescreen usually has an aspect ratio of 16:9 or 16/9 = 1.78. The camera apect ratio is set up automatically. However if you are using split screen effects you may need to change the

camera aspect ratio.

Related

CreatePtrKeyMapArray

Helper function: returns pointer to a keymap for _IrrAddFPSCamera.

#Include <au3Irrlicht2.au3>
___CreatePtrKeyMapArray(ByRef \$keyStruct, \$i_kForward =
\$KEY_KEY_W, \$i_kBackward = \$KEY_KEY_S, \$i_kLeft =
\$KEY_KEY_A, \$i_kRight = \$KEY_KEY_D, \$i_kJump = \$KEY_SPACE)

Parameters

	Any variable which is then returned as a a keymap forIrrAddFPSCamera.
\$i_kForward	Key for forward movement, default is \$KEY_KEY_W
\$i_kBackward	Key for backward movement, default is \$KEY_KEY_S
\$i_kLeft	Key for left movement, default is \$KEY_KEY_A
\$i_kRight	Key for right movement, default is \$KEY_KEY_D
\$i_kJump	Key for jumping, default is \$KEY_SPACE

Return Value

Success: Pointer to Dllstruct containing key struct useable by _IrrAddFpsCamera. Failure: False

Remarks

\$keyStruct can be set to '0' after call of _IrraddFPSCamera to delete the DIIstruct. Usage with defaults creates WASD keys, e.g.: _IrrAddFPSCamera (..., __CreatePtrKeyMapArray(\$keyStruct), 4, ...)

Related

<u>_IrraddFPSCamera</u>

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $keyStruct
local $camera = _IrrAddFPSCamera( 0, 150.0, 0.1, -1, __
          _CreatePtrKeyMapArray($keyStruct), 4 )
$keyStruct = 0
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $sceneNode = IrrAddMeshToScene( $mesh )
_IrrSetNodePosition($sceneNode, 0, 0, 5)
local $texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
_IrrSetWindowCaption("Move with mouse + WASD keys - quit with ALT-
F4!")
WHILE _IrrRunning()
  _IrrBeginScene(50, 50, 50)
  _IrrDrawScene()
  _IrrEndScene()
WEND
_IrrStop()
```

Copy to Clipboard

IrrGetCollisionGroupFromMesh

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetCollisionGroupFromMesh(\$h_Mesh, \$h_Node, \$i_Frame = 0)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

<u>_IrrGetCollisionGroupFromComplexMesh</u>, <u>_IrrGetCollisionGroupFromBox</u>, <u>_IrrGetCollisionGroupFromTerrain</u>, <u>_IrrRemoveCollisionGroup</u>, <u>_IrrCreateCombinedCollisionGroup</u>

IrrGetCollisionGroupFromComplexMesh

Creates an optimized triangle selection group from a large complex mesh like a map.

#Include <au3Irrlicht2.au3>
_IrrGetCollisionGroupFromComplexMesh(\$h_Mesh, \$h_Node, \$i_Frame =
0)

Parameters

\$h_Mesh	Handle of mesh the node was created from.
\$h_Node	Handle of the node to create a selector from.
\$i_Frame	[optional] Number of mesh frame to use.

Return Value

success: Handle to a selector object failure: False

Remarks

The returned triangle selection group can then be used in collision functions to collide objects against this node.

Related

_IrrGetCollisionGroupFromMesh, _IrrGetCollisionGroupFromBox, _IrrGetCollisionGroupFromTerrain, _IrrRemoveCollisionGroup, _IrrCreateCombinedCollisionGroup

Example

```
#include "au3Irrlicht2.au3"
```

```
_IrrStart()
```

```
_IrrAddZipFile( "./media/map-20kdm2.pk3", $IRR_IGNORE_CASE,
$IRR_IGNORE_PATHS )
$meshBSP = _IrrGetMesh( "20kdm2.bsp" )
local $nodeBSP = _IrrAddMeshToSceneAsOcttree( $meshBSP )
```

```
local $nodeCamera = _IrrAddFPSCamera()
_IrrSetNodePosition( $nodeCamera, 1750, 149, 1369 )
```

```
WHILE _IrrRunning()

_IrrBeginScene(50, 50, 50)

_IrrDrawScene()

_IrrEndScene()

WEND
```

_IrrStop()

Copy to Clipboard

IrrGetCollisionGroupFromBox

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetCollisionGroupFromBox(\$h_Node)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

<u>_IrrGetCollisionGroupFromMesh</u>, <u>_IrrGetCollisionGroupFromComplexMesh</u>, <u>_IrrGetCollisionGroupFromTerrain</u>, <u>_IrrRemoveCollisionGroup</u>, <u>_IrrCreateCombinedCollisionGroup</u>

IrrCreateCombinedCollisionGroup

[todo]

#Include <au3Irrlicht2.au3>
_IrrCreateCombinedCollisionGroup()

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

<u>_IrrGetCollisionGroupFromMesh</u>, <u>_IrrGetCollisionGroupFromComplexMesh</u>, <u>_IrrGetCollisionGroupFromBox</u>, <u>_IrrGetCollisionGroupFromTerrain</u>, <u>_IrrRemoveCollisionGroup</u>

IrrAddCollisionGroupToCombination

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddCollisionGroupToCombination(\$h_CombinedCollisionGroup,
\$h_CollisionGroup)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetCollisionPoint

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetCollisionPoint(\$a_StartVector, \$a_EndVector, \$h_CollisionGroup,
byRef \$a_CollisionVector)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetCollisionNodeFromCamera

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetCollisionNodeFromCamera(\$h_Camera)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetCollisionNodeFromRay

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetCollisionNodeFromRay(byRef \$h_StartVector, byRef \$h_EndVector)

Parameters

[param1]	[explanation] [moreTextForParam1]	
----------	--------------------------------------	--

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetCollisionNodeFromScreenCoordinates

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetCollisionNodeFromScreenCoordinates(\$i_X, \$i_Y)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetScreenCoordinatesFrom3DPosition

Screen co-ordinates are returned for the position of the specified 3D coordinates.

#Include <au3Irrlicht2.au3>
_IrrGetScreenCoordinatesFrom3DPosition(ByRef \$i_ScreenX, ByRef
\$i_ScreenY, \$a_3DPositionVector)

Parameters

	Variables which will contain coordinates after call of the function.
\$a_3DPositionVector	1D array with three elements for x, y, z values of a position in space.

Return Value

Success: True and sets passed \$i_ScreenX and \$i_ScreenY Failure: False and @error = 1

Remarks

Screen co-ordinates are returned for the position of the specified 3D coordinates as if an object were drawn at them on the screen, this is ideal for drawing 2D bitmaps or text around or on your 3D object on the screen for example in the HUD of an aircraft.

Related

```
<u>_IrrGet3DPositionFromScreenCoordinates</u>,
<u>_IrrGet2DPositionFromScreenCoordinates</u>
```

Example

```
local $nodeCam = _IrrAddFPSCamera( $IRR_NO_OBJECT, 100.0, 0.1 )
_IrrSetNodePosition( $nodeCam, 80,0,0 )
_IrrSetCameraTarget( $nodeCam, 0,0,0 )
```

\$vect[1] = 35 ; move the vector to just over the nodes head

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 25)
_IrrDrawScene()
```

; using 3D coordinate get x/y position of this coordinate; it appears on the screen

_IrrGetScreenCoordinatesFrom3DPosition(\$x, \$y, \$vect) ; draw the name of the model over the head of the model _Irr2DFontDraw (\$BitmapFont, "ZUMLIN", \$x-15, \$y-8, \$x+35, \$y)

```
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrGet3DPositionFromScreenCoordinates

[todo]

#Include <au3Irrlicht2.au3>
_IrrGet3DPositionFromScreenCoordinates(\$i_X, \$i_Y, ByRef
\$a_Vector3df, \$h_Camera, \$f_NormalX=0.0, \$f_NormalY=0.0,
\$f_NormalZ=1.0, \$f_DistanceFromOrigin=0.0)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGet2DPositionFromScreenCoordinates

[todo]

#Include <au3Irrlicht2.au3>
_IrrGet2DPositionFromScreenCoordinates(\$i_X, \$i_Y, ByRef \$f_X, ByRef
\$f_Y, \$h_Camera)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetChildCollisionNodeFromRay

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetChildCollisionNodeFromRay(\$h_Node, \$i_Mask, \$i_Recurse,
\$a_StartVector, \$a_EndVector)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetChildCollisionNodeFromPoint

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetChildCollisionNodeFromPoint(\$h_Node, \$i_Mask, \$i_Recurse,
\$a_PointVector)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetNodeAndCollisionPointFromRay

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetNodeAndCollisionPointFromRay(\$a_StartVector, \$a_EndVector,
ByRef \$h_Node, ByRef \$f_PosX, ByRef \$f_PosY, ByRef \$f_PosZ, ByRef
\$f_NormalX, ByRef \$f_NormalY, ByRef \$f_NormalZ, \$i_ID = 0,
\$h_RootNode = \$IRR_NO_OBJECT)

Parameters

maramii	[explanation] [moreTextForParam1]
---------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetDistanceBetweenNodes

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetDistanceBetweenNodes(\$h_NodeA, \$h_NodeB)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAreNodesIntersecting

[todo]

#Include <au3Irrlicht2.au3>
_IrrAreNodesIntersecting(\$h_NodeA, \$h_NodeB)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrIsPointInsideNode

[todo]

#Include <au3Irrlicht2.au3>
_IrrIsPointInsideNode(\$h_NodeA, \$f_X, \$f_Y, \$f_Z)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetCollisionResultPosition

[todo]

#Include <au3Irrlicht2.au3>

_IrrGetCollisionResultPosition(\$h_Selector, ByRef \$a_EllipsoidPosition, ByRef \$a_EllipsoidRadius, ByRef \$a_Velocity, ByRef \$a_Gravity, \$f_SlidingSpeed, ByRef \$a_OutPosition, ByRef \$a_OutHitPosition, ByRef \$i_OutFalling)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddZipFile

Adds a zip archive to the filing system allowing to load files out of the zip file.

#Include <au3Irrlicht2.au3>
_IrrAddZipFile(\$s_Zipfile, \$i_IgnoreCase, \$i_IgnorePaths)

Parameters

\$s_ZipFile	Path to the zipfile (or pk3 file)
\$i_IgnoreCase	Should be one of the following values: \$IRR_USE_CASE \$IRR_IGNORE_CASE
\$i_IgnorePaths	Ignore paths allows you to simply use the filename without the path, the filename should always be unique in the archive when using this option. The value should be one of the following: \$IRR_USE_PATHS \$IRR_IGNORE_PATHS

Return Value

Success: True Failure: False

Remarks

Files inside the .zip can be opened as if they were in the current working directory.

Common pk3 files are simply zip files.

Related

None.

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
_IrrAddZipFile( "./media/map-20kdm2.pk3", $IRR_IGNORE_CASE,
$IRR_IGNORE_PATHS )
$BSPMesh = _IrrGetMesh( "20kdm2.bsp" )
local $BSPNode = _IrrAddMeshToSceneAsOcttree( $BSPMesh )
local $Camera = _IrrAddFPSCamera()
_IrrSetNodePosition( $camera, 1750, 149, 1369 )
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
_IrrStop()
```

Copy to Clipboard

IrrChangeWorkingDirectory

[todo]

#Include <au3Irrlicht2.au3>
_IrrChangeWorkingDirectory(\$s_WorkingDir)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetWorkingDirectory

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetWorkingDirectory()

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

getGuiEvt

Helper function: returns value of \$i_Element inside a GuiEvent-structure.

#Include <au3Irrlicht2.au3>
__getGuiEvt(\$p_GUIEvent, \$i_Element = \$EVT_GUI_IID)

Parameters

<pre>\$p_GUIEvent</pre>	A pointer as returned from _IrrReadGUIEvent.
\$i_Element	[optional] Event type to return: \$EVT_GUI_IID - ID of GUI element as integer \$EVT_GUI_IEVENT - Event type from enum \$IRR_EGUI_EVENT_TYPE \$EVT_GUI_IX - X as integer EVT_GUI_IY - Y as integer

Return Value

Success: Value of selected event type Failure: Returns False and sets @error = 1

Remarks

[todo]

Related

_IrrReadGUIEvent

IrrGUIClear

[todo]

#Include <au3Irrlicht2.au3>
_IrrGUIClear()

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGUIEvents

[todo]

#Include <au3Irrlicht2.au3>
_IrrGUIEvents(\$i_EventsForGUI)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGUIEventAvailable

[todo]

#Include <au3Irrlicht2.au3>
_IrrGUIEventAvailable()

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrReadGUIEvent

[todo]

#Include <au3Irrlicht2.au3>
_IrrReadGUIEvent()

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGUIRemove

[todo]

#Include <au3Irrlicht2.au3>
_IrrGUIRemove(\$h_Widget)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGUIGetText

[todo]

#Include <au3Irrlicht2.au3>
_IrrGUIGetText(\$h_Widget)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGUISetText

[todo]

#Include <au3Irrlicht2.au3>
_IrrGUISetText(\$h_Widget, \$s_Text)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGUISetFont

[todo]

#Include <au3Irrlicht2.au3>
_IrrGUISetFont(\$h_Font)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGUISetColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrGUISetColor(\$i_Element, \$i_Red, \$i_Green, \$i_Blue, \$i_Alpha)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddWindow

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddWindow(\$s_Title, \$i_TopX, \$i_TopY, \$i_BottomX, \$i_BottomY,
\$i_Modal, \$h_Parent = 0)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddStaticText

Creates a static text object on the Graphical User Interface.

#Include <au3Irrlicht2.au3>
_IrrAddStaticText(\$s_Text, \$i_TopX, \$i_TopY, \$i_BottomX, \$i_BottomY,
\$i_Border, \$i_WordWrap, \$h_Parent = 0)

Parameters

\$s_Text	String that contains the text you want to display.
\$i_TopX	Top X position of a box in which the text is drawn
\$i_TopY	Top Y position of a box in which the text is drawn
\$i_BottomX	Bottom X position of a box in which the text is drawn
\$i_BottomY	Bottom Y position of a box in which the text is drawn
\$i_Border	Border is used to draw a visible box around the text, its value should be either of: \$IRR_GUI_NO_BORDER \$IRR_GUI_BORDER
\$i_WordWrap	Word wrap is used to define whether text is to be wrapped around into a second line when it fills the width of the text box, its value should be either of: \$IRR_GUI_NO_WRAP \$IRR_GUI_WRAP
\$h_Parent	[optional] Parent defines the parent object of this window. This can be ommited if the object has no parent.

Return Value

Success: Pointer to the static text Object Failure: False

Remarks

This function simply displays the specifed text in the specified box.

Related

_IrrDrawGUI

Example

#include "au3Irrlicht2.au3"

```
_IrrStart( $IRR_EDT_OPENGL, 800, 600, $IRR_BITS_PER_PIXEL_32, _
$IRR_WINDOWED, $IRR_SHADOWS, $IRR_CAPTURE_EVENTS,
$IRR_VERTICAL_SYNC_ON )
```

```
_IrrAddStaticText( "Hello AU3-World ;-)", 8, 8, 200, 20,
$IRR_GUI_NO_BORDER, $IRR_GUI_NO_WRAP)
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawGUI()
_IrrEndScene()
WEND
```

```
_IrrStop()
```

```
Copy to Clipboard
```

IrrAddButton

[todo]

```
#Include <au3Irrlicht2.au3>
_IrrAddButton($i_TopX, $i_TopY, $i_BottomX, $i_BottomY, $i_ID,
$s_Text = "", $s_TextTip = "", $h_Parent = 0)
```

Parameters

Inaramii	[explanation] [moreTextForParam1]	
----------	--------------------------------------	--

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddScrollBar

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddScrollBar(\$i_Horizontal, \$i_TopX, \$i_TopY, \$i_BottomX,
\$i_BottomY, \$i_ID, \$i_CurrentValue, \$i_MaxValue, \$h_Parent = 0)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddListBox

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddListBox(\$i_TopX, \$i_TopY, \$i_BottomX, \$i_BottomY, \$i_ID,
\$i_Background, \$h_Parent=0)

Parameters

	[explanation] [moreTextForParam1]	
--	--------------------------------------	--

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddListBoxItem

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddListBoxItem(\$h_ListBox, \$s_Text)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSelectListBoxItem

[todo]

#Include <au3Irrlicht2.au3>
_IrrSelectListBoxItem(\$h_ListBox, \$i_Index)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddEditBox

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddEditBox(\$s_Text, \$i_TopX, \$i_TopY, \$i_BottomX, \$i_BottomY,
\$i_ID, \$i_Border, \$i_Password = \$IRR_GUI_NOT_PASSWORD,
\$h_Parent = 0)

Parameters

[ovplanation]	
[param1] [explanation] [moreTextForParam1]	

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddCheckBox

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddCheckBox(\$s_Text, \$i_TopX, \$i_TopY, \$i_BottomX, \$i_BottomY,
\$i_ID, \$i_Checked, \$h_Parent = 0)

Parameters

Ingramii	[explanation] [moreTextForParam1]	
----------	--------------------------------------	--

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddImage

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddImage(\$h_Texture, \$i_X, \$i_Y, \$i_UseAlpha, \$i_ID, \$h_Parent = 0)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddFileOpen

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddFileOpen(\$s_Title, \$i_ID, \$i_Modal, \$h_Parent = 0)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetLastSelectedFile

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetLastSelectedFile()

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

getKeyEvt

helper function: returns value of \$i_Element inside a keyEvent-structure.

#Include <au3Irrlicht2.au3>
___getKeyEvt(\$p_KeyEvent, \$i_Element = \$EVT_KEY_IKEY)

Parameters

<pre>\$p_KeyEvent</pre>	A pointer as returned from _IrrReadKeyEvent.		
\$i_Element	[optional] Event type to return: \$EVT_KEY_IKEY - ID of pressed key (see remarks). \$EVT_KEY_IDIRECTION - Direction value - can be either \$IRR_KEY_DOWN or \$IRR_KEY_UP \$EVT_KEY_IFLAGS - Bits are set in this parameter to specify whether the shift or control key was keydown at the time the key action occured.		

Return Value

Success: Value of selected event element.

Remarks

\$p_KeyEvent is a pointer as returned from _IrrReadKeyEvent.

Key code table for \$EVT_KEY_IKEY:

Left mouse button	\$KEY_RBUTTON	F n b
Control-break		N n b

\$KEY_CANCEL	processing	\$KEY_MBUTTON	ן) b ח
\$KEY_XBUTTON1	Windows 2000/XP: X1 mouse button	\$KEY_XBUTTON2	V 2 X b
\$KEY_BACK	BACKSPACE key	\$KEY_TAB	Т
\$KEY_CLEAR	Clear key	\$KEY_RETURN	E
\$KEY_SHIFT	SHIFT key	\$KEY_CONTROL	C
\$KEY_MENU	ALT key	\$KEY_PAUSE	F
\$KEY_CAPITAL	CAPS Lock key	\$KEY_KANA	ll n
\$KEY_HANGUEL	IME Hanguel mode (maintained For compatibility use KEY_HANGUL)	AKEY_HANGUL	lî n
\$KEY_JUNJA	IME Junja mode	\$KEY_FINAL	ll n
\$KEY_HANJA	IME Hanja mode	\$KEY_KANJI	ll n
\$KEY_ESCAPE	ESC key	\$KEY_CONVERT	11
\$KEY_NONCONVERT	IME nonconvert	\$KEY_ACCEPT	11
\$KEY_MODECHANGE	IME mode change request	\$KEY_SPACE	S
\$KEY_PRIOR	PAGE UP key	\$KEY_NEXT	F C
\$KEY_END	End key	\$KEY_HOME	F
\$KEY_LEFT	Left ARROW key	\$KEY_UP	L A k

\$KEY_RIGHT	Right ARROW key	\$KEY_DOWN	С А к
\$KEY_SELECT	Select key	\$KEY_PRINT	P
\$KEY_EXECUT	EXECUTE key	\$KEY_SNAPSHOT	P S
\$KEY_INSERT	INS key	\$KEY_DELETE	
\$KEY_HELP	HELP key	\$KEY_KEY_0	0
\$KEY_KEY_1	1 key		
\$KEY_KEY_9	9 key	\$KEY_KEY_A	Α
\$KEY_KEY_B	B key		
\$KEY_KEY_Z	Z key	\$KEY_LWIN	L k (< k
\$KEY_RWIN	Right Win key (Natural keyboard)	\$KEY_APPS	A k k
\$KEY_SLEEP	Computer Sleep key	\$KEY_NUMPAD0	N k k
\$KEY_NUMPAD1	Numeric keypad 1 key		
\$KEY_NUMPAD9	Numeric keypad 9 key	\$KEY_MULTIPLY	N
\$KEY_ADD	Add key	\$KEY_SEPARATOR	S k
\$KEY_SUBTRACT	Subtract key	\$KEY_DECIMAL	C k
\$KEY_DIVIDE	Divide key	\$KEY_F1	F
\$KEY_F2	F2 key		
\$KEY_F24	F24 key	\$KEY_NUMLOCK	N k

\$KEY_SCROLL	SCROLL Lock key	\$KEY_LSHIFT	L k
\$KEY_RSHIFT	Right SHIFT key	\$KEY_LCONTROL	L C k
\$KEY_RCONTROL	Right CONTROL key	\$KEY_LMENU	L k
\$KEY_RMENU	Right MENU key	\$KEY_PLUS	F (·
\$KEY_COMMA	Comma Key (,)	\$KEY_MINUS	N (-
\$KEY_PERIOD	Period Key (.)	\$KEY_ATTN	Α
\$KEY_CRSEL	CrSel key	\$KEY_EXSEL	E
\$KEY_EREOF	Erase Eof key	\$KEY_PLAY	F
\$KEY_ZOOM	Zoom key	\$KEY_PA1	F
\$KEY_OEM_CLEAR	Clear key	\$KEY_KEY_CODES_COUNT	T a tł O tł

Related

_IrrReadKeyEvent, _IrrKeyEventAvailable

Example

#include "au3Irrlicht2.au3"

local \$pKeyEvent

```
; enable event capturing:
_IrrStart($IRR_EDT_DIRECT3D9, 800, 600, $IRR_BITS_PER_PIXEL_32,
```

getMouseEvt

helper function: returns value of \$i_Element inside a MouseEventstructure.

#Include <au3Irrlicht2.au3>
___getMouseEvt(\$p_MouseEvent, \$i_Element = \$EVT_MOUSE_IACTION)

Parameters

<pre>\$p_MouseEvent</pre>	A pointer as returned from _IrrReadMouseEvent.
\$i_Element	[optional] Event type to return: \$EVT_MOUSE_IACTION - ID of mouse action (see remarks). \$EVT_MOUSE_FDELTA - Amount of movement of the mouse wheel (> 0 means wheel up, < 0 means wheel down). \$EVT_MOUSE_IX - Horizontal screen coordinate at which the event took place. \$EVT_MOUSE_IY - Vertical screen coordinate at which the event took place.

Return Value

Success: Value of selected event element.

Remarks

\$p_MouseEvent is a pointer as returned from _IrrReadMouseEvent.

Mouse actions table for **\$EVT_MOUSE_IACTION**:

\$IRR_EMIE_LMOUSE_PRESSED_DOV	/N Left mouse button pressed
	Right mouse button

\$IRR_EMIE_RMOUSE_PRESSED_DOWN	
\$IRR_EMIE_MMOUSE_PRESSED_DOWN	Middle mouse button pressed
\$IRR_EMIE_LMOUSE_LEFT_UP	Left mouse button released
\$IRR_EMIE_RMOUSE_LEFT_UP	Right mouse button released
	Middle mouse button released
	Mouse was moved horizontal and/or vertical
	Mouse wheel was moved up or down

Related

_IrrReadMouseEvent, _IrrMouseEventAvailable

Example

```
#include "au3Irrlicht2.au3"
```

local \$pMouseEvent

```
; enable event capturing:
_IrrStart($IRR_EDT_DIRECT3D9, 800, 600, $IRR_BITS_PER_PIXEL_32,
```

```
$IRR_WINDOWED, $IRR_NO_SHADOWS,
$IRR_CAPTURE_EVENTS)
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 25)
```

; process all available mouse events:

```
while _IrrMouseEventAvailable()
      $pMouseEvent = _IrrReadMouseEvent()
      ; check for mousewheel event, report up or down movement inside this
window title:
      if __getMouseEvt($pMouseEvent, $EVT_MOUSE_IACTION) =
$IRR EMIE MOUSE WHEEL then
        if __getMouseEvt($pMouseEvent, $EVT_MOUSE_FDELTA) > 0
Then
          _IrrSetWindowCaption("Mousewheel up")
        Else
          _IrrSetWindowCaption("Mousewheel down")
        EndIf
      endif
    wend
  _IrrEndScene()
WEND
Copy to Clipboard
```

IrrKeyEventAvailable

Determine if there are any keystrokes waiting to be read.

#Include <au3Irrlicht2.au3>
_IrrKeyEventAvailable()

Parameters

None.

Return Value

Success: True if there are keystrokes, otherwise False.

Remarks

Event capturing needs to be enabled before with _IrrStart or _IrrStartAdvanced!

Related

<u>_IrrReadKeyEvent, __getKeyEvt, _IrrStart</u>

Example

#include "au3Irrlicht2.au3"

local \$pKeyEvent

```
; enable event capturing:
_IrrStart($IRR_EDT_DIRECT3D9, 800, 600, $IRR_BITS_PER_PIXEL_32,
_______$IRR_WINDOWED, $IRR_NO_SHADOWS,
$IRR_CAPTURE_EVENTS)
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 25)
; process all available key events and check for ESC:
while _IrrKeyEventAvailable()
        $pKeyEvent = _IrrReadKeyEvent()
        if __getKeyEvt($pKeyEvent, $EVT_KEY_IKEY) = $KEY_ESCAPE
then _IrrStop()
        WEnd
        _IrrEndScene()
WEND
Copy to Clipboard
```

IrrReadKeyEvent

Read a key event from the Irrlicht window.

#Include <au3Irrlicht2.au3>
_IrrReadKeyEvent()

Parameters

None.

Return Value

success: Pointer of a key event. failure: False

Remarks

The properties of the returned key event are readable with the helper function ___getKeyEvt.

Related

_IrrKeyEventAvailable, __getKeyEvt

Example

#include "au3Irrlicht2.au3"

local \$pKeyEvent

IrrMouseEventAvailable

Determine if there are any mouse events waiting to be read.

#Include <au3Irrlicht2.au3>
_IrrMouseEventAvailable()

Parameters

None.

Return Value

Success: True if there are mouse events, otherwise False.

Remarks

Event capturing needs to be enabled before with _IrrStart or _IrrStartAdvanced!

Related

IrrReadMouseEvent, __getMouseEvt, _IrrStart

Example

#include "au3Irrlicht2.au3"

local \$pMouseEvent

; enable event capturing: _IrrStart(\$IRR_EDT_DIRECT3D9, 800, 600, \$IRR_BITS_PER_PIXEL_32,

```
$IRR_WINDOWED, $IRR_NO_SHADOWS,
$IRR_CAPTURE_EVENTS)
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 25)
```

```
; process all available mouse events:
while _IrrMouseEventAvailable()
    $pMouseEvent = _IrrReadMouseEvent()
```

; check for mousewheel event, report up or down movement inside this window title:

```
if __getMouseEvt($pMouseEvent, $EVT_MOUSE_IACTION) =
$IRR_EMIE_MOUSE_WHEEL then
```

```
if __getMouseEvt($pMouseEvent, $EVT_MOUSE_FDELTA) > 0
```

Then

```
_IrrSetWindowCaption("Mousewheel up")
```

Else

```
_IrrSetWindowCaption("Mousewheel down")
EndIf
```

endif

wend

```
_IrrEndScene()
WEND
```

Copy to Clipboard

IrrReadMouseEvent

Read a key event from the Irrlicht window.

#Include <au3Irrlicht2.au3>
_IrrReadMouseEvent()

Parameters

None.

Return Value

success: Pointer of a mouse event. failure: False

Remarks

The properties of the returned mouse event are readable with the helper function ___getMouseEvt.

Related

_IrrMouseEventAvailable, __getMouseEvt

Example

#include "au3Irrlicht2.au3"

local \$pMouseEvent

; enable event capturing:

_IrrStart(\$IRR_EDT_DIRECT3D9, 800, 600, \$IRR_BITS_PER_PIXEL_32,

```
$IRR_WINDOWED, $IRR_NO_SHADOWS,
$IRR_CAPTURE_EVENTS)
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 25)
```

; process all available mouse events: while _IrrMouseEventAvailable() \$pMouseEvent = _IrrReadMouseEvent()

; check for mousewheel event, report up or down movement inside this window title:

```
if __getMouseEvt($pMouseEvent, $EVT_MOUSE_IACTION) =
$IRR_EMIE_MOUSE_WHEEL then
```

if __getMouseEvt(\$pMouseEvent, \$EVT_MOUSE_FDELTA) > 0

Then

```
_IrrSetWindowCaption("Mousewheel up")
```

Else

```
_IrrSetWindowCaption("Mousewheel down")
```

EndIf

endif

wend

```
_IrrEndScene()
WEND
```

Copy to Clipboard

IrrSetMousePosition

Sets relative position of the mouse pointer and returns relative position before this change.

#Include <au3Irrlicht2.au3>
_IrrSetMousePosition(ByRef \$f_XPos, ByRef \$f_YPos)

Parameters

\$f_XPos	Fractional value for new horizontal position (0-1).
\$f_YPos	Fractional value for new vertical position (0-1).

Return Value

success: True and sets \$f_XPos and \$f_YPos to relative position where the mouse was before (both 0-1). failure: False

Remarks

This function works independent from the resolution of current Irrlicht display. Expected and returned values are fractional values, where 0/0 is top left and 1/1 bottom right of the Irrlicht display.

Related

_IrrReadMouseEvent, _IrrGetAbsoluteMousePosition

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
_IrrAddSkyDomeToScene(_IrrGetTexture("./media/au3irr2_logo.jpg"), 16,
16, 1, 2)
local $nodeCamera = _IrrAddCamera( 0,0,5, 0,0,0 )
local $X, $Y, $camY
```

_IrrHideMouse() WHILE _IrrRunning() _IrrBeginScene(0, 0, 25)

; center mouse cursor \$X = .5 \$Y = .5 _IrrSetMousePosition(\$X, \$Y)

```
; add any mouse movement to the cam-Y position ...
$camY += (0.5 - $Y)
```

```
; ... and reset it slowly back to centre of the display:
if $camY > 0 then $camY -= 0.005
if $camY < 0 then $camY += 0.005
_IrrSetCameraTarget($nodeCamera, 0, $camY, 0)
```

```
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrGetAbsoluteMousePosition

Get the absolute mouse X and Y position

#Include <au3Irrlicht2.au3>
_IrrGetAbsoluteMousePosition()

Parameters

None.

Return Value

success: 1D Array containing absolute mouse position. \$Array[0] = X position of the mouse \$Array[1] = Y position of the mouse failure: Empty 1D Array and Set @error 1

Remarks

None.

Related

Example

#include <au3Irrlicht2.au3>

Global \$aGAMP

```
_IrrStart()
```

While _IrrRunning()

```
$aGAMP = _IrrGetAbsoluteMousePosition()
_IrrSetWindowCaption("Absolute Mouse Poition - X: " & $aGAMP[0] & "
Y: " & $aGAMP[1])
```

WEnd

_IrrStop()

Copy to Clipboard

IrrHideMouse

Hides the mouse pointer

#Include <au3Irrlicht2.au3>
_IrrHideMouse()

Parameters

None.

Return Value

Success: True Failure: False

Remarks

None

Related

<u>_IrrShowMouse</u>, <u>_IrrDisplayMouse</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $camera = _IrrAddCamera(2,2,2, 0,0,0 )
local $time = TimerInit()
WHILE _IrrRunning()

if int(mod( (TimerDiff($time) / 1000), 2 )) = true Then
    _IrrHideMouse()
Else
    _IrrShowMouse()
EndIf
    _IrrBeginScene(50, 50, 50)
    _IrrDrawScene()
    _IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrShowMouse

Shows the mouse pointer

#Include <au3Irrlicht2.au3>
_IrrShowMouse()

Parameters

None.

Return Value

Success: True Failure: False

Remarks

None

Related

_IrrHideMouse, _IrrDisplayMouse

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $camera = _IrrAddCamera(2,2,2, 0,0,0 )
local $time = TimerInit()
WHILE _IrrRunning()

if int(mod( (TimerDiff($time) / 1000), 2 )) = true Then
    _IrrHideMouse()
Else
    _IrrShowMouse()
EndIf
    _IrrBeginScene(50, 50, 50)
    _IrrDrawScene()
    _IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrDisplayMouse

Hide or show the mouse pointer while it is within the Irrlicht display.

#Include <au3Irrlicht2.au3>
_IrrDisplayMouse(\$i_HideShow)

Parameters

\$i_HideShow True shows and False hides the mouse pointer.

Return Value

Success: True Failure: False

Remarks

There are two functions available to simply hide or show the mouse: IrrHideMouse and IrrShowMouse.

Related

<u>IrrShowMouse</u>, <u>IrrHideMouse</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $camera = _IrrAddCamera(0,0,0, 0,0,0 )
```

```
local $time = TimerInit()
WHILE _IrrRunning()
```

_IrrDisplayMouse(int(mod((TimerDiff(\$time) / 1000), 2)))

```
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrAddLight

Adds a light node into scene to naturally illuminate your scene.

#Include <au3Irrlicht2.au3>
_IrrAddLight(\$h_parentNode, \$f_X, \$f_Y, \$f_Z, \$f_Red, \$f_Green,
\$f_Blue, \$f_Size)

Parameters

\$h_parentNode	Handle of the node to attach the light to. \$IRR_NO_PARENT attaches to the root node of the scene.
\$f_X, \$f_Y, \$f_Z	Coordinates of the light in the scene
\$f_Red, \$f_Green, \$f_Blue	Intensity of the light. Red/green/blue are fractional values from 0 to 1!
\$f_Size	Radius of effect of the light

Return Value

Success: Handle of light node in the scene Failure: False

Remarks

When using shadows you probably only want one or two lights - they can be time consuming.

Related

_IrrAddNodeShadow, _IrrSetAmbientLight

Example

#include "au3Irrlicht2.au3"

```
_IrrStart( $IRR_EDT_OPENGL, 800, 600, $IRR_BITS_PER_PIXEL_32, _
$IRR_WINDOWED, $IRR_SHADOWS, $IRR_IGNORE_EVENTS,
$IRR_VERTICAL_SYNC_ON )
```

```
local $nodeSphere = _IrrAddSphereSceneNode(4, 32)
_IrrSetNodeMaterialTexture( $nodeSphere,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
```

```
local $nodeLight = _IrrAddLight($IRR_NO_PARENT, 0,50,0, 0.9,0.9,0.9,
1000.0)
_IrrAddFlyCircleAnimator($nodeLight, 0, 0, 0, 500, 0.001)
```

```
local $nodeCamera = _IrrAddCamera(-5,5,-6, 0,0,0 )
```

```
WHILE _IrrRunning()
_IrrBeginScene(100, 100, 100)
_IrrDrawScene()
_IrrEndScene()
WEND
```

```
_IrrStop()
Copy to Clipboard
```

IrrSetAmbientLight

Sets the ambient lighting level across entire scene.

#Include <au3Irrlicht2.au3>
_IrrSetAmbientLight(\$f_Red, \$f_Green, \$f_Blue)

Parameters

\$i_Red, \$i_Green, \$i Blue	Colour values for ambient lighting. Red/green/blue are fractional values from 0 to 1!
------------------------------------	------------------------------------------------------------------------------------------

Return Value

Success: True Failure: False

Remarks

Ambient light illuminates all surfaces in the scene uniformly. This is usually a low value to increase the overall lighting level. It should never be greater than the brightness of the darkest area of your scene, it can however reduce the number of lights you need in the scene.

Related

_IrrAddLight, _IrrSetLightAmbientColor

Example

#include "au3Irrlicht2.au3"

```
_IrrStart( $IRR_EDT_OPENGL, 800, 600, $IRR_BITS_PER_PIXEL_32, _
$IRR_WINDOWED, $IRR_SHADOWS, $IRR_IGNORE_EVENTS,
$IRR_VERTICAL_SYNC_ON )
```

```
local $nodeSphere = _IrrAddSphereSceneNode(4, 32)
_IrrSetNodeMaterialTexture( $nodeSphere,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
```

```
local $nodeLight = _IrrAddLight($IRR_NO_PARENT, 0,50,0, 0.9,0.9,0.9,
1000.0)
_IrrSetAmbientLight(8, 0, 0)
IrrAddFlyCircleAnimator($nodeLight, 0, 0, 0, 500, 0.001)
```

```
local $nodeCamera = _IrrAddCamera(-5,5,-6, 0,0,0 )
```

```
WHILE _IrrRunning()

_IrrBeginScene(100, 100, 100)

_IrrDrawScene()

_IrrEndScene()

WEND
```

```
_IrrStop()
Copy to Clipboard
```

IrrSetLightAmbientColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetLightAmbientColor(\$h_Light, \$f_Red, \$f_Green, \$f_Blue)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetLightAttenuation

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetLightAttenuation(\$h_Light, \$f_Red, \$f_Green, \$f_Blue)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetLightDiffuseColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetLightDiffuseColor(\$h_Light, \$f_Red, \$f_Green, \$f_Blue)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetLightFalloff

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetLightFalloff(\$h_Light, \$f_Falloff)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetLightInnerCone

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetLightInnerCone(\$h_Light, \$f_InnerCone)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetLightOuterCone

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetLightOuterCone(\$h_Light, \$f_OuterCone)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetLightType

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetLightType(\$h_Light, \$i_Type)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetNodeAmbientColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetNodeAmbientColor(\$h_Node, \$i_Color)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetNodeDiffuseColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetNodeDiffuseColor(\$h_Node, \$i_Color)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetNodeSpecularColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetNodeSpecularColor(\$h_Node, \$i_Color)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetNodeEmissiveColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetNodeEmissiveColor(\$h_Node, \$i_Color)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetNodeColorByVertex

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetNodeColorByVertex(\$h_Node, \$i_ColorMaterial)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrMaterialVertexColorAffects

[todo]

#Include <au3Irrlicht2.au3>
_IrrMaterialVertexColorAffects(\$h_Material, \$i_AffectedProperty)

Parameters

[param1]	[explanation] [moreTextForParam1]	
----------	--------------------------------------	--

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetMaterialBlend

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetMaterialBlend(\$h_Material, \$i_SrcBlend, \$i_DstBlend)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrMaterialSetShininess

[todo]

#Include <au3Irrlicht2.au3>
_IrrMaterialSetShininess(\$h_Material, \$f_Shininess)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrMaterialSetSpecularColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrMaterialSetSpecularColor(\$h_Material, \$i_Alpha, \$i_Red, \$i_Green,
\$i_Blue)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrMaterialSetDiffuseColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrMaterialSetDiffuseColor(\$h_Material, \$i_Alpha, \$i_Red, \$i_Green,
\$i_Blue)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrMaterialSetAmbientColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrMaterialSetAmbientColor(\$h_Material, \$i_Alpha, \$i_Red, \$i_Green,
\$i_Blue)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrMaterialSetEmissiveColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrMaterialSetEmissiveColor(\$h_Material, \$i_Alpha, \$i_Red, \$i_Green,
\$i_Blue)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrMaterialSetMaterialTypeParam

[todo]

#Include <au3Irrlicht2.au3>
_IrrMaterialSetMaterialTypeParam(\$h_Material, \$f_Param)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrXEffectsStart

[todo]

#Include <au3Irrlicht2.au3>
_IrrXEffectsStart(\$i_Vsm=\$IRR_OFF, \$i_SoftShadows=\$IRR_OFF,
\$iBitdepth32=\$IRR_OFF)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrXEffectsAddShadowToNode

[todo]

#Include <au3Irrlicht2.au3>
_IrrXEffectsAddShadowToNode(\$h_Node, \$i_FilterType=\$EFT_NONE,
\$i_shadowType=\$ESM_BOTH)

Parameters

	[explanation] [moreTextForParam1]	Ī
--	--------------------------------------	---

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrXEffectsAddShadowLight

[todo]

#Include <au3Irrlicht2.au3>

_IrrXEffectsAddShadowLight(\$i_ShadowDimen, \$f_PosX, \$f_PosY,

\$f_PosZ, \$f_TargetX, \$f_TargetY, \$f_TargetZ, \$f_R, \$f_G, \$f_B, \$f_Alpha,

\$f_LightNearDist, \$f_LightFarDist, \$f_AngleDegrees)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrXEffectsSetAmbientColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrXEffectsSetAmbientColor(\$i_R, \$i_G, \$i_B, \$i_Alpha)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrXEffectsSetClearColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrXEffectsSetClearColor(\$i_R, \$i_G, \$i_B, \$i_Alpha)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrXEffectsSetShadowLightPosition

[todo]

#Include <au3Irrlicht2.au3>
_IrrXEffectsSetShadowLightPosition(\$i_Index, \$f_PosX, \$f_PosY,
\$f_PosZ)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrXEffectsEnableDepthPass

[todo]

#Include <au3Irrlicht2.au3>
_IrrXEffectsEnableDepthPass(\$i_Enable)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrXEffectsAddPostProcessingFromFile

[todo]

#Include <au3Irrlicht2.au3>
_IrrXEffectsAddPostProcessingFromFile(\$s_Name, \$i_Enable=0)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrXEffectsAddNodeToDepthPass

[todo]

#Include <au3Irrlicht2.au3>
_IrrXEffectsAddNodeToDepthPass(\$h_Node)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrXEffectsSetPostProcessingUserTexture

[todo]

#Include <au3Irrlicht2.au3>
_IrrXEffectsSetPostProcessingUserTexture(\$h_Texture)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetNodeName

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetNodeName(\$h_Node)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetNodeName

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetNodeName(\$h_Node, \$s_Name)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetNodeMesh

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetNodeMesh (\$h_Node)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetMaterialCount

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetMaterialCount(\$h_Node)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetMaterial

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetMaterial(\$h_Node, \$i_Material)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetNodeMaterialTexture

Applys a texture to a node in the scene.

#Include <au3Irrlicht2.au3>
_IrrSetNodeMaterialTexture(\$h_Node, \$h_Texture, \$i_Index)

Parameters

\$h_Node	Handle of a node in the scene
\$h_Texture	Handle of a texture object
\$i_Index	Material index number of the material layer, usually 0 or 1.

Return Value

Success: True Failure: False

Remarks

How the texture is applied across the surface of the node will depend on the texturing coordinates in each of the vectors of the mesh and how they are plotted across the surface of the texture.

Some nodes can have several textures applied to them to create special material effects.

Related

_IrrSetNodeMaterialFlag, _IrrSetNodeMaterialType

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $camera = _IrrAddCamera(2,2,2, 0,0,0 )
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $sceneNode = _IrrAddMeshToScene( $mesh )
local $texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
WHILE _IrrRunning()
  _IrrBeginScene(50, 50, 50)
  _IrrDrawScene()
  _IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrSetNodeMaterialFlag

Sets material properies of a node that will effect its appearance on the screen.

#Include <au3Irrlicht2.au3>
_IrrSetNodeMaterialFlag(\$h_Node, \$i_Type, \$i_Flag)

Parameters

\$h_Node	Handle of a node in the scene
\$i_Type	Material type is one of the following properties: \$IRR_EMF_WIREFRAME - Render as wireframe outline \$IRR_EMF_GOURAUD_SHADING - Render smoothly across polygons \$IRR_EMF_LIGHTING - Material is effected by lighting \$IRR_EMF_ZBUFFER - Enable z buffer \$IRR_EMF_ZWRITE_ENABLE - Can write as well as read z buffer \$IRR_EMF_BACK_FACE_CULLING - Cull polygons facing away \$IRR_EMF_BILINEAR_FILTER - Enable bilinear filtering \$IRR_EMF_TRILINEAR_FILTER - Enable trilinear filtering \$IRR_EMF_ANISOTROPIC_FILTER - Reduce blur in distant textures \$IRR_EMF_FOG_ENABLE - Enable fogging in the distance \$IRR_EMF_NORMALIZE_NORMALS - Use when scaling dynamically lighted models
\$i_Flag	Switches selected property on or off: \$IRR_ON \$IRR_OFF

Return Value

Success: True Failure: False

Remarks

None.

Related

_IrrSetNodeMaterialTexture, _IrrSetNodeMaterialType

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $camera = _IrrAddCamera(2,2,2, 0,0,0)
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $sceneNode = _IrrAddMeshToScene( $mesh )
local $texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
WEND
_IrrStop()
```

Copy to Clipboard

IrrSetNodeMaterialType

Set the way that materials are applied to a scene node

#Include <au3Irrlicht2.au3>
_IrrSetNodeMaterialType(\$h_Node, \$i_Type)

Parameters

\$h_Node	Handle of a scene node
\$i_Type	Material property to apply to the scene node (see table in remarks)

Return Value

success: True failure: False

Remarks

Valid material properties:

\$IRR_EMT_SOLID	Stanc
\$IRR_EMT_SOLID_2_LAYER	2 blei
\$IRR_EMT_LIGHTMAP	2 text ignor
\$IRR_EMT_LIGHTMAP_ADD	as modu
\$IRR_EMT_LIGHTMAP_M2	as 2 for
\$IRR_EMT_LIGHTMAP_M4	as for br

\$IRR_EMT_LIGHTMAP_LIGHTING	2 text suppr
\$IRR_EMT_LIGHTMAP_LIGHTING_M2	as 2 for
\$IRR_EMT_LIGHTMAP_LIGHTING_M4	as 4 for
\$IRR_EMT_DETAIL_MAP	2 bler secor from
\$IRR_EMT_SPHERE_MAP	make
\$IRR_EMT_REFLECTION_2_LAYER	a reflettextur
\$IRR_EMT_TRANSPARENT_ADD_COLOR	a trar textuı color
\$IRR_EMT_TRANSPARENT_ALPHA_CHANNEL	a trar textui
\$IRR_EMT_TRANSPARENT_ALPHA_CHANNEL_REF	a trar textur alpha not bl etc.
\$IRR_EMT_TRANSPARENT_VERTEX_ALPHA	a trar alpha
\$IRR_EMT_TRANSPARENT_REFLECTION_2_LAYER	a trar textuı map.
\$IRR_EMT_NORMAL_MAP_SOLID	A soli color, adde IrrAd Only verte:
\$IRR_EMT_NORMAL_MAP_TRANSPARENT_ADD_COLOR	as that s the da

\$IRR_EMT_NORMAL_MAP_TRANSPARENT_VERTEX_ALPHA	as that u
\$IRR_EMT_PARALLAX_MAP_SOLID	simila realis surfa map 1 verte:
\$IRR_EMT_PARALLAX_MAP_TRANSPARENT_ADD_COLOR	as that s the da
\$IRR_EMT_PARALLAX_MAP_TRANSPARENT_VERTEX_ALPHA	as that u

Related

_IrrSetNodeMaterialTexture, _IrrSetNodeMaterialFlag

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCamera = _IrrAddCamera(2,2,2, 0,0,0 )
local $nodeCube = _IrrAddCubeSceneNode(2) ; _IrrAddMeshToScene( $mesh
)
_IrrSetNodeMaterialTexture( $nodeCube,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0 )
```

_IrrSetNodeMaterialFlag(\$nodeCube, \$IRR_EMF_LIGHTING, \$IRR_OFF) _IrrSetNodeMaterialType(\$nodeCube, \$IRR_EMT_SPHERE_MAP)

_IrrAddRotationAnimator(\$nodeCube, 0.1, 0.1, 0.1)

```
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrSetNodePosition

Moves the node to the specified position

#Include <au3Irrlicht2.au3>
_IrrSetNodePosition(\$h_Node, \$f_X, \$f_Y, \$f_Z)

Parameters

\$h_Node	Handle of a scene node
\$f_X, \$f_Y, \$f_Z	X, Y, Z values of new position

Return Value

Success: True Failure: False

Remarks

None

Related

<u>_IrrGetNodePosition</u>, <u>_IrrGetNodeRotation</u>, <u>_IrrSetNodeRotation</u>, <u>_IrrGetNodeAbsolutePosition</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCube = _IrrAddCubeSceneNode(5)
_IrrSetNodeMaterialTexture( $nodeCube,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeCube, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
_IrrSetNodePosition($nodeCube, 5, 5, 5)
local $nodeCamera = _IrrAddCamera(0, 0, 0, 5, 5, 5)
```

```
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrSetNodeRotation

Rotate a node to the specified orientation through its X, Y and Z axis

#Include <au3Irrlicht2.au3>
_IrrSetNodeRotation(\$h_Node, \$f_X, \$f_Y, \$f_Z)

Parameters

\$h_Node	Handle of a scene node
\$f_X, \$f_Y, \$f_Z	Values of rotation along X, Y, Z axes in degrees (0-360)

Return Value

Success: True Failure: False

Remarks

None

Related

<u>_IrrGetNodePosition</u>, <u>_IrrSetNodePosition</u>, <u>_IrrGetNodeRotation</u>,

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCamera = _IrrAddCamera(10, 0, 0, 0, 0, 0, 0)
local $nodeCube = _IrrAddCubeSceneNode(5)
_IrrSetNodeMaterialTexture( $nodeCube,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeCube, $IRR_EMF_LIGHTING, $IRR_OFF )
_IrrSetNodeRotation($nodeCube, 90, 45, 45 )
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrSetNodeScale

Sets the scale of the scene node

#Include <au3Irrlicht2.au3>
_IrrSetNodeScale(\$h_Node, \$f_X, \$f_Y, \$f_Z)

Parameters

\$h_Node	Handle of a scene node
\$f_X, \$f_Y, \$f_Z	Scaling factors for X, Y, Z axes.

Return Value

Success: True Failure: False

Remarks

None

Related

_IrrSetNodePosition, _IrrSetNodeRotation

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCamera = _IrrAddCamera(5,10,10, 0,0,0)
local $nodeTest = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
_IrrSetNodeScale($nodeTest, 0.5, 1, 1.5)
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrDebugDataVisible

Displays debugging data around a node (typically the bounding box around edges of the node).

#Include <au3Irrlicht2.au3>
_IrrDebugDataVisible(\$h_Node, \$i_Visible = \$EDS_BBOX)

Parameters

\$h_Node	Handle of a scene node.	
\$i_Visible	Type of debugging information (not all of them are supported on all node types): \$EDS_OFF: No Debugging \$EDS_BBOX: Bounding Box \$EDS_NORMALS: Normals \$EDS_SKELETON: Skeleton \$EDS_MESH_WIRE_OVERLAY: Wireframe \$EDS_HALF_TRANSPARENCY: Transparency \$EDS_BBOX_BUFFERS: Bounding Box Buffers \$EDS_FULL: Everything	

Return Value

None.

Remarks

None.

Related

None.

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $camera = _IrrAddFPSCamera()
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $nodeTest = _IrrAddMeshToScene( $mesh )
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
_IrrSetNodePosition($nodeTest, 0, 0, 8 )
```

```
_IrrDebugDataVisible($nodeTest, $EDS_FULL)
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

IrrGetNodePosition

Returns array with position coordinates of a scene node.

#Include <au3Irrlicht2.au3>
_IrrGetNodePosition(\$h_Node)

Parameters

\$h_Node Handle of a scene node

Return Value

success: 0-based array with three elements for X, Y, Z coordinates. failure: False

Remarks

None.

Related

<u>_IrrSetNodePosition</u>, <u>_IrrGetNodeRotation</u>, <u>_IrrSetNodeRotation</u>,

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeTest = _IrrAddTestSceneNode()
local $nodeCamera = _IrrAddFPSCamera()
_IrrSetNodePosition($nodeCamera, 0, 0, -50)
```

local \$aVector3df

```
WHILE _IrrRunning()
_IrrBeginScene( 0, 0, 50 )
_IrrDrawScene()
```

```
$aVector3df = _IrrGetNodePosition($nodeCamera)
_IrrSetWindowCaption("Camera position (x/y/z): " & _
    int($aVector3df[0]) & " / " & int($aVector3df[1]) & " / " &
int($aVector3df[2]))
```

```
_IrrEndScene()
WEND
```

_IrrStop()

IrrGetNodeAbsolutePosition

Get the absoloute position of the node in the scene.

#Include <au3Irrlicht2.au3>
_IrrGetNodeAbsolutePosition(\$h_Node)

Parameters

\$h_Node Handle of a scene node

Return Value

success: 0-based array with three elements for X, Y, Z coordinates. failure: False

Remarks

The absolute postion includes the position changes of all of the nodes parents too.

Related

_IrrGetNodePosition, _IrrSetNodePosition, _IrrGetNodeRotation, _IrrSetNodeRotation

IrrGetNodeRotation

Returns array with rotation values of a scene node.

#Include <au3Irrlicht2.au3>
_IrrGetNodeRotation(\$h_Node)

Parameters

\$h_Node Handle of a scene node

Return Value

success: 0-based array with three elements for X, Y, Z rotation. failure: False

Remarks

None.

Related

<u>_IrrGetNodePosition</u>, <u>_IrrSetNodePosition</u>, <u>_IrrSetNodeRotation</u>,

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeTest = _IrrAddTestSceneNode()
local $nodeCamera = _IrrAddFPSCamera()
_IrrSetNodePosition($nodeCamera, 0, 0, -50)
```

local \$aVector3df

```
WHILE _IrrRunning()
_IrrBeginScene( 0, 0, 50 )
_IrrDrawScene()
```

```
$aVector3df = _IrrGetNodeRotation($nodeCamera)
_IrrSetWindowCaption("Camera rotation (x/y/z): " & _
    int($aVector3df[0]) & " / " & int($aVector3df[1]) & " / " &
int($aVector3df[2]))
```

```
_IrrEndScene()
WEND
```

_IrrStop()

IrrGetJointNode

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetJointNode(\$h_Node, \$s_Joint)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddChildToParent

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddChildToParent(\$h_ChildNode, \$h_ParentNode)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetNodeFirstChild

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetNodeFirstChild(\$h_Node, ByRef \$h_Position)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetNodeNextChild

Get the next child node of this node, returns 0 if there is no child.

#Include <au3Irrlicht2.au3>
_IrrGetNodeNextChild(\$h_Node, ByRef \$h_Position)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrIsNodeLastChild

[todo]

#Include <au3Irrlicht2.au3>
_IrrIsNodeLastChild(\$h_Node, ByRef \$h_Position)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddNodeShadow

Adds shadows to a node that are cast across other nodes in the scene.

#Include <au3Irrlicht2.au3>
_IrrAddNodeShadow(\$h_Node, \$h_mesh = 0)

Parameters

\$h_Node	Handle of a node in the scene	
\$h_mesh	[optional] Handle of mesh casting the shadow (0 uses mesh of h_Node, see remarks)	

Return Value

Success: True Failure: False

Remarks

_IrrAddShadows will only work when shadowing has been activated with _IrrStart or _IrrStartAdvanced.

You should analyse the performance of your scene carefully when using this function as it can have a significant effect on your frame rate.

You can supply a different mesh to the one used to display the node, this shadow mesh could be a much lower resoloution than that used for your model thereby improving performance.

_IrrAddNodeShadow does NOT work with buildin nodes types _IrrAddTestSceneNode, _IrrAddCubeSceneNode, and _IrrAddSphereSceneNode. If you need a cube or a sphere casting shadows, create a node from a loaded cube or sphere mesh as workaround.

Related

_IrrStart, _IrrStartAdvanced, _IrrSetShadowColor, _IrrAddLight, _IrrSetAmbientLight

Example

#include "au3Irrlicht2.au3"

```
_IrrStart( $IRR_EDT_OPENGL, 800, 600, $IRR_BITS_PER_PIXEL_32, _
$IRR_WINDOWED, $IRR_SHADOWS, $IRR_IGNORE_EVENTS,
$IRR_VERTICAL_SYNC_ON )
```

```
local $texture = _IrrGetTexture(".\media\au3irr2_logo.jpg")
```

```
local $meshReceive = _IrrAddHillPlaneMesh( "Plane", 8, 8, 1, 1)
local $nodeReceive = _IrrAddMeshToScene( $meshReceive )
_IrrSetNodeMaterialTexture( $nodeReceive, $texture, 0)
```

```
local $meshCast = _IrrGetMesh(".\media\sphere.obj")
local $nodeCast = _IrrAddMeshToScene($meshCast)
_IrrSetNodeMaterialTexture( $nodeCast, $texture, 0)
_IrrAddFlyCircleAnimator($nodeCast, 1, 3, 1, 3, 0.0005)
```

```
local $nodeLight = _IrrAddLight($IRR_NO_PARENT, 0,50,0, 0.9,0.3,0.3,
1000.0)
_IrrAddNodeShadow($nodeCast)
local $nodeCamera = _IrrAddCamera(-5,5,-6, 0,0,0 )
```

```
WHILE _IrrRunning()
_IrrBeginScene(100, 100, 100)
_IrrDrawScene()
```

_IrrEndScene() WEND

_IrrStop()

IrrSetNodeVisibility

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetNodeVisibility(\$h_Node, \$i_Visible)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrRemoveNode

Removes a node from the scene deleting it.

#Include <au3Irrlicht2.au3>
_IrrRemoveNode(\$h_Node)

Parameters

\$h_Node Handle of a scene node

Return Value

Success: True Failure: False

Remarks

None

Related

None.

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $camera = _IrrAddCamera(2,2,2, 0,0,0 )
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $sceneNode = _IrrAddMeshToScene( $mesh )
local $texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
local $time = TimerInit()
WHILE _IrrRunning()
  _IrrBeginScene(50, 50, 50)
  _IrrDrawScene()
  _IrrEndScene()
  if $sceneNode <> 0 AND TimerDiff($time) > 3000 then
    _IrrRemoveNode($sceneNode)
    _IrrRemoveTexture($texture); no longer needed
    sceneNode = 0
  EndIf
WEND
_IrrStop()
Copy to Clipboard
```

IrrRemoveAllNodes

[todo]

#Include <au3Irrlicht2.au3>
_IrrRemoveAllNodes()

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetNodeID

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetNodeID(\$h_Node, \$i_ID)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

CreateParticleSettings

Helper function: Creates particle settings as required from _IrrAddParticleEmitter.

#Include <au3Irrlicht2.au3>

__CreateParticleSettings(\$minBoxX, \$minBoxY, \$minBoxZ, \$maxBoxX, \$maxBoxY, \$maxBoxZ, \$directionX, \$directionY, \$directionZ, \$minParticlesSecond, \$maxParticlesSecond, \$minStartColorR, \$minStartColorG, \$minStartColorB, \$maxStartColorR, \$maxStartColorG, \$maxStartColorB, \$minLifetime, \$maxLifetime, \$minStartSizeX, \$minStartSizeY, \$maxStartSizeX, \$maxStartSizeY, \$maxAngleDegrees)

Parameters

\$minBoxX, \$minBoxY, \$minBoxZ	Minimal positions of a a box in space inside which the position of a particle is randomly created.
\$maxBoxX, \$maxBoxY, \$maxBoxZ	Maximal positions of a a box in space inside which the position of a particle is randomly created.
\$directionX, \$directionY, \$directionZ	Define a direction into which the particles will be ejected as the animation plays.
\$minParticlesSecond, \$maxParticlesSecond	A range defining the minimum and maximum number of particles that will be created each second.
<pre>\$minStartColorR, \$minStartColorG, \$minStartColorB, \$maxStartColorR, \$maxStartColorG, \$maxStartColorB</pre>	Although particles can be textured by texturing the particle system node, these can be used to apply a range that tints the color of the particles.

\$minLifetime, \$maxLifetime	How long the partilce will live, long lifespans can create very large numbers of particles.
\$minStartSizeX, \$minStartSizeY, \$maxStartSizeX, \$maxStartSizeY	The minimum and maximum start sizes for the particles.
\$maxAngleDegrees	The maximum number of degrees that the ejected particles will deviate from the defined direction.

Return Value

None.

Remarks

None

Related

_IrrAddParticleSystemToScene, _IrrAddParticleEmitter

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $SmokeEmitter = __CreateParticleSettings( -7.0, 0, -7.0, 7.0, 1.0, 7.0, 0,
0.04, 0, 80, 100, __
```

255, 255, 255, 255, 255, 255, _ 800, 2000, 15.0, 15.0, 15.0, 15.0, 15)

_IrrAddParticleEmitter(\$particleSystem, \$SmokeEmitter)

```
local $ParticleTexture = _IrrGetTexture( "./media/ParticleGrey.bmp" )
_IrrSetNodeMaterialTexture( $particleSystem, $ParticleTexture, 0 )
_IrrSetNodeMaterialFlag( $particleSystem, $IRR_EMF_LIGHTING,
$IRR_OFF )
_IrrSetNodeMaterialType ( $particleSystem,
$IRR_EMT_TRANSPARENT_ADD_COLOR )
```

```
local $nodeCamera = _IrrAddCamera( 80,0,0, 20,40,0 )
```

```
WHILE _IrrRunning()
_IrrBeginScene( 0,0,50 )
_IrrDrawScene()
_IrrEndScene()
```

WEND

_IrrStop()

IrrAddParticleEmitter

Adds a particle emitter to a particle system.

#Include <au3Irrlicht2.au3>
_IrrAddParticleEmitter(\$h_ParticleSystem, \$a_Settings)

Parameters

\$h_ParticleSystem	Handle of particle system the emitter shall be attached to.
\$a_Settings	Array with particle emitter settings created withCreateParticleSettings.

Return Value

success: Handle of the created particle emitter failure: False

Remarks

The emitter creates particles and controls how they move and when they are to be removed.

Related

<u>_IrrAddParticleSystemToScene</u>, <u>__CreateParticleSettings</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

local \$SmokeEmitter = __CreateParticleSettings(-7.0, 0, -7.0, 7.0, 1.0, 7.0, 0,
0.04, 0, 80, 100, __

255, 255, 255, 255, 255, 255, <u></u>800, 2000, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0,

_IrrAddParticleEmitter(\$particleSystem, \$SmokeEmitter)

local \$ParticleTexture = _IrrGetTexture("./media/ParticleGrey.bmp")
_IrrSetNodeMaterialTexture(\$particleSystem, \$ParticleTexture, 0)
_IrrSetNodeMaterialFlag(\$particleSystem, \$IRR_EMF_LIGHTING,
\$IRR_OFF)
_IrrSetNodeMaterialType (\$particleSystem,

```
$IRR_EMT_TRANSPARENT_ADD_COLOR )
```

local \$nodeCamera = _IrrAddCamera(80,0,0, 20,40,0)

```
WHILE _IrrRunning()
_IrrBeginScene( 0,0,50 )
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

IrrAddAnimatedMeshSceneNodeEmitter

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddAnimatedMeshSceneNodeEmitter(\$h_ParticleSystem, \$h_Node,
\$b_UseNormals, \$f_NormalModifier, \$b_FromAllVertices, \$a_Settings)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddRotationAffector

Adds a an affector to a particle system rotating the particles.

#Include <au3Irrlicht2.au3>
_IrrAddRotationAffector(\$h_ParticleSystem, \$f_SpeedX, \$f_SpeedY,
\$f_SpeedZ, \$f_PivotX, \$f_pivotY, \$f_pivotZ)

Parameters

\$h_ParticleSystem	Handle of the particle system the created affector is attached to.
\$f_SpeedX, \$f_SpeedY, \$f_SpeedZ	Set the speed in degrees per second in all 3 dimensions.
\$f_PivotX, \$f_pivotY, \$f_pivotZ	Set the point that particles will rotate around.

Return Value

success: Handle of the created affector. failure: false

Remarks

This affector modifies the positions of the particles and attracts them to a specified point at a specified speed per second.

Related

<u>_IrrAddParticleSystemToScene</u>, <u>_IrrSetParticleAffectorEnable</u>, <u>_IrrRemoveAffectors</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

; add particle system with default emitter to the irrlicht scene manager local \$particleSystem =

_IrrAddParticleSystemToScene(\$IRR_DEFAULT_EMITTER)
local \$ParticleTexture = _IrrGetTexture("./media/ParticleGrey.bmp")

; setup affector rotating particles around all three axes with particles attracted to 30/0/0:

_IrrAddRotationAffector (\$particleSystem, 50.0, -120.0, 50.0, 30.0,0.0,0.0)

_IrrSetNodeMaterialTexture(\$particleSystem, \$ParticleTexture, 0)
_IrrSetNodeMaterialFlag(\$particleSystem, \$IRR_EMF_LIGHTING,
\$IRR_OFF)
_IrrSetNodeMaterialType (\$particleSystem,

\$IRR_EMT_TRANSPARENT_ADD_COLOR)

local \$nodeCamera = _IrrAddCamera(80,0,0, 0,50,50)

```
WHILE _IrrRunning()
_IrrBeginScene( 0,0,50 )
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

IrrAddFadeOutParticleAffector

Adds a fade out affector to a particle system gradually fading particles out so they are invisible when they are deleted.

#Include <au3Irrlicht2.au3>
_IrrAddFadeOutParticleAffector(\$h_ParticleSystem, \$i_FadeFactor, \$i_Red,
\$i_Green, \$i_Blue)

Parameters

\$h_ParticleSystem	Handle of the particle system the created affector is attached to.
\$i_FadeFactor	Milliseconds the fade out effect will take place.
\$i_Red, \$i_Green, \$i_Blue	Values of the colour the particles are faded to (0- 255)

Return Value

success: Handle of the created affector. failure: false

Remarks

The fade out affector fades the particles out as they come to the end of their lifespan and stops them 'popping' out of existance. This creates a convincing effect for fire and smoke in particular.

Related

_IrrAddParticleSystemToScene, _IrrSetParticleAffectorEnable,

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
; add particle system with default emitter to the irrlicht scene manager
local $particleSystem =
_IrrAddParticleSystemToScene($IRR_DEFAULT_EMITTER)
local $ParticleTexture = _IrrGetTexture( "./media/ParticleGrey.bmp" )
; setup this affector for a simple flashing-out effect
_IrrAddFadeOutParticleAffector( $particleSystem, 250, 255, 255, 0)
_IrrSetNodeMaterialTexture( $particleSystem, $ParticleTexture, 0 )
IrrSetNodeMaterialFlag( $particleSystem, $IRR EMF LIGHTING,
$IRR OFF)
_IrrSetNodeMaterialType ( $particleSystem,
$IRR EMT TRANSPARENT ADD COLOR)
local $nodeCamera = _IrrAddCamera( 80,0,0, 20,40,0 )
WHILE _IrrRunning()
  _IrrBeginScene(0,0,50)
  _IrrDrawScene()
  _IrrEndScene()
WEND
_IrrStop()
 Copy to Clipboard
```

IrrAddGravityParticleAffector

Adds a gravity affector to a particle system gradually pulling the particles in the direction of the effect.

```
#Include <au3Irrlicht2.au3>
_IrrAddGravityParticleAffector($h_ParticleSystem, $f_X, $f_Y, $f_Z,
$i_TimeForceLost = 1000)
```

Parameters

\$h_ParticleSystem	Handle of the particle system the created affector is attached to.
\$f_X, \$f_Y, \$f_Z	Set the direction and force of gravity in all 3 dimensions.
	[optional] Set the time in milliseconds when the gravity force is totally lost. At that point the particle does not move any more.

Return Value

success: Handle of the created affector. failure: false

Remarks

The gravity affector is adding a small amount of velocity to the particles each frame. Although its called a gravity affector it can be used to push the particles in any direction so you can have drifting smoke bubbling fountains, to make a wind effect and have the particles drift off to the side, etc.

Related

<u>_IrrAddParticleSystemToScene</u>, <u>_IrrSetParticleAffectorEnable</u>, <u>_IrrRemoveAffectors</u>

Example

#include "au3Irrlicht2.au3"
_IrrStart()
; add particle system with default emitter to the irrlicht scene manager
local \$particleSystem =
_IrrAddParticleSystemToScene(\$IRR_DEFAULT_EMITTER)
local \$ParticleTexture = _IrrGetTexture("./media/ParticleGrey.bmp")
; setup this affector to let the particles drift off
_IrrAddGravityParticleAffector(\$particleSystem, -0.4, 0, 0.8, 3000)
_IrrSetNodeMaterialTexture(\$particleSystem, \$ParticleTexture, 0)
_IrrSetNodeMaterialFlag(\$particleSystem, \$IRR_EMF_LIGHTING,
\$IRR_OFF)
_IrrSetNodeMaterialType (\$particleSystem,
\$IRR_EMT_TRANSPARENT_ADD_COLOR)

```
WHILE _IrrRunning()
_IrrBeginScene( 0,0,50 )
_IrrDrawScene()
_IrrEndScene()
WEND
```

```
_IrrStop()
```

IrrAddParticleAttractionAffector

Adds an affector to a particle system attracting particles to a specified point at a specified speed.

#Include <au3Irrlicht2.au3>
_IrrAddParticleAttractionAffector(\$h_ParticleSystem, \$f_X, \$f_Y, \$f_Z,
\$f_Speed = 1, \$i_Attract = \$IRR_ATTRACT, \$b_AffectX = true,
\$b_AffectY = true, \$b_AffectZ = true)

Parameters

\$h_ParticleSystem	Handle of the particle system the created affector is attached to.
\$f_X, \$f_Y, \$f_Z	Set the point that particles will attract to.
\$f_Speed	[optional] Speed in units per second, to attract to the specified point.
\$i_Attract	[optional] Set whether or not the particles are attracting or detracting. Values are: \$IRR_ATTRACT - particles are attracting. IRR_REPEL - particles are detracting.
\$b_AffectX, \$b_AffectY, \$b_AffectZ	[optional] Set whether or not this will affect particles in the X, Y, Z direction.

Return Value

success: Handle of the created affector. failure: false

Remarks

None.

Related

<u>_IrrAddParticleSystemToScene</u>, <u>_IrrSetParticleAffectorEnable</u>, <u>_IrrRemoveAffectors</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

; add particle system with default emitter to the irrlicht scene manager local \$particleSystem =

```
_IrrAddParticleSystemToScene($IRR_DEFAULT_EMITTER)
local $ParticleTexture = _IrrGetTexture( "./media/ParticleGrey.bmp" )
```

; setup this affector to detract particles downwards _IrrAddParticleAttractionAffector (\$particleSystem, 0,100.0,0.0, 80.0, \$IRR_REPEL)

_IrrSetNodeMaterialTexture(\$particleSystem, \$ParticleTexture, 0)
_IrrSetNodeMaterialFlag(\$particleSystem, \$IRR_EMF_LIGHTING,
\$IRR_OFF)

_IrrSetNodeMaterialType (\$particleSystem, \$IRR_EMT_TRANSPARENT_ADD_COLOR)

local \$nodeCamera = _IrrAddCamera(80,0,0, 0,-20,0)

```
WHILE _IrrRunning()
_IrrBeginScene( 0,0,50 )
_IrrDrawScene()
_IrrEndScene()
```

WEND

_IrrStop()

Copy to Clipboard

IrrAddStopParticleAffector

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddStopParticleAffector(\$h_ParticleSystem, \$i_Time, \$h_Emitter)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddParticlePushAffector

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddParticlePushAffector(\$h_ParticleSystem, \$f_X, \$f_Y, \$f_Z,
\$f_SpeedX, \$f_SpeedY, \$f_SpeedZ, \$f_Far, \$f_Near, \$f_Column,
\$i_Distant)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddColorMorphAffector

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddColorMorphAffector(\$h_ParticleSystem, \$a_ParticleColors,
\$a_ParticleTimes, \$b_Smooth)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddSplineAffector

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddSplineAffector(\$h_ParticleSystem, \$tVectors, \$f_Speed,
\$f_Tightness, \$f_Attraction, \$b_DeleteAtEnd)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrRemoveAffectors

Removes all affectors from a particle system.

#Include <au3Irrlicht2.au3>
_IrrRemoveAffectors(\$h_ParticleSystem)

Parameters

\$h_ParticleSystem Handle of an Irrlicht particle system.

Return Value

success: true failure: false

Remarks

You might use this if you want to change the direction or strength of the wind for example.

Related

_IrrAddParticleSystemToScene, _IrrSetParticleAffectorEnable

Example

#include "au3Irrlicht2.au3"

_IrrStart()

; add particle system with default emitter to the irrlicht scene manager local \$particleSystem =

_IrrAddParticleSystemToScene(\$IRR_DEFAULT_EMITTER)
local \$ParticleTexture = _IrrGetTexture("./media/ParticleGrey.bmp")

; add two affectors to the particle system:

_IrrAddRotationAffector (\$particleSystem, 50.0, -120.0, 50.0, 30.0,0.0,0.0) _IrrAddFadeOutParticleAffector(\$particleSystem, 100, 255, 0, 0)

_IrrSetNodeMaterialTexture(\$particleSystem, \$ParticleTexture, 0)
_IrrSetNodeMaterialFlag(\$particleSystem, \$IRR_EMF_LIGHTING,
\$IRR_OFF)

_IrrSetNodeMaterialType (\$particleSystem, \$IRR_EMT_TRANSPARENT_ADD_COLOR)

local \$nodeCamera = _IrrAddCamera(80,0,0, 0,50,50)

```
local $time = TimerInit()
WHILE _IrrRunning()
```

if TimerDiff(\$time) > 8000 then _IrrRemoveAffectors(\$particleSystem)

```
_IrrBeginScene( 0,0,50 )
_IrrDrawScene()
_IrrEndScene()
```

WEND

_IrrStop()

Copy to Clipboard

IrrSetParticleEmitterMinParticlesPerSecond

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetParticleEmitterMinParticlesPerSecond(\$h_ParticleEmitter, \$i_Min)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetParticleEmitterMaxParticlesPerSecond

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetParticleEmitterMaxParticlesPerSecond(\$h_ParticleEmitter, \$i_Max)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetParticleEmitterMinStartColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetParticleEmitterMinStartColor(\$h_ParticleEmitter, \$i_Red, \$i_Green,
\$i_Blue)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetParticleEmitterMaxStartColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetParticleEmitterMaxStartColor(\$h_ParticleEmitter, \$i_Red, \$i_Green,
\$i_Blue)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetParticleAffectorEnable

Enables or disables an Irrlicht affector.

#Include <au3Irrlicht2.au3>
_IrrSetParticleAffectorEnable(\$h_ParticleAffector, \$b_Enabled)

Parameters

h_ParticleAffector Handle of an Irrlicht particle affector	
\$i_Enabled	\$IRR_ON (or true) enables the affector, \$IRR_OFF (or false) disables it.

Return Value

success: true failure: false

Remarks

None

Related

_IrrAddParticleSystemToScene, _IrrRemoveAffectors

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
; add particle system with default emitter to the irrlicht scene manager local $particleSystem =
```

```
_IrrAddParticleSystemToScene($IRR_DEFAULT_EMITTER)
local $ParticleTexture = _IrrGetTexture( "./media/ParticleGrey.bmp" )
```

```
local $affector = _IrrAddRotationAffector ( $particleSystem, 50.0, -120.0,
50.0, 30.0,0.0,0.0 )
```

```
_IrrSetNodeMaterialTexture( $particleSystem, $ParticleTexture, 0 )
_IrrSetNodeMaterialFlag( $particleSystem, $IRR_EMF_LIGHTING,
$IRR_OFF )
```

```
_IrrSetNodeMaterialType ( $particleSystem,
$IRR_EMT_TRANSPARENT_ADD_COLOR )
```

```
local $nodeCamera = _IrrAddCamera( 80,0,0, 0,50,50)
```

```
local $time = TimerInit()
local $enabled = true
WHILE _IrrRunning()
```

```
; enable/disable affector every 2 seconds:
if TimerDiff($time) > 2000 then
    $enabled = NOT $enabled
    _IrrSetParticleAffectorEnable($affector, $enabled)
    $time = TimerInit()
EndIf
```

```
_IrrBeginScene( 0,0,50 )
_IrrDrawScene()
_IrrEndScene()
```

WEND

_IrrStop()

Copy to Clipboard

IrrSetFadeOutParticleAffectorTime

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetFadeOutParticleAffectorTime(\$h_ParticleAffector, \$f_Time)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetGravityParticleAffectorDirection

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetGravityParticleAffectorDirection(\$h_ParticleAffector, \$f_X, \$f_Y,
\$f_Z)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetCenterOfEffect

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetCenterOfEffect(\$h_ParticleAffector, \$f_X, \$f_Y, \$f_Z)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetRootSceneNode

Get the scenes root node, all scene nodes are children of this node.

#Include <au3Irrlicht2.au3>
_IrrGetRootSceneNode()

Parameters

None.

Return Value

Success: Handle of root node in the scene Failure: False and @error 1

Remarks

[todo]

Related

_IrrAddMeshToScene

IrrGetMesh

Loads the specified mesh ready to be added to the scene.

#Include <au3Irrlicht2.au3>
_IrrGetMesh(\$s_MeshFile)

Parameters

\$s_MeshFile Filename of the mesh object to load

Return Value

Success: Handle of the loaded mesh object Failure: False

Remarks

The Irrlicht engine supports a wide range of mesh types.

Static objects:

	42 4
Irrlicht static meshes (.irrmesh, r/w)	3D Studio meshes (.3ds, r)
Alias Wavefront Maya (.obj, r/w)	Lightwave Objects (.lwo, r)
COLLADA 1.4 (.xml, .dae, r/w)	OGRE meshes (.mesh, r)
My3DTools 3 (.my3D, r)	LMTools (.lmts, r)
Quake 3 levels (.bsp, r)	DeleD (.dmf, r)
FSRad oct (.oct, r)	Cartography shop 4 (.csm, r)
STL 3D files (.stl, r/w)	PLY 3D files (.ply, r/w)

Animated objects:

Microsoft DirectX (.x, r) (binary & text,

B3D files (.b3d, r, skeleton)	skeleton)
Milkshape (.ms3d, r, skeleton)	Quake 3 models (.md3, r, morph)

Related

_IrrAddMeshToScene, _IrrRemoveMesh

Example

#include "au3Irrlicht2.au3"

_IrrStart()

local \$camera = _IrrAddCamera(2,2,2, 0,0,0)

```
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $sceneNode = _IrrAddMeshToScene( $mesh )
```

```
local $texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
```

```
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrCreateMesh

Create a new mesh from lists of vertices and indices.

#Include <au3Irrlicht2.au3>
_IrrCreateMesh(\$s_MeshName, \$tVertexArray, \$a_Indices)

Parameters

\$s_MeshName	String name for the newly created mesh object.
\$tVertexArray	Vertex array struct as created with <u>CreateVertStruct</u> or returned from _IrrGetMeshVertices.
\$a Indices	3D-array with list of indices as returned from _IrrGetMeshIndices or created e.g. with DIM \$aIndices[indicesNumber] = [0,1,4, 1,2,4,]

Return Value

Success: Handle to the newly created mesh object Failure: False and set @error:

@error 1 : either AutoIt DIICall or IrrCreateMesh call failed

@error 2 : \$tVertexArray param is not a Struct

@error 2 : \$a_Indices param is not an Array

Remarks

You must supply a list of vertices inside a vertex array struct and an array of indices that refer to these vertices.

The indices are taken in groups of three joining up the dots defined by the verticies and forming a collection of triangles.

Related

<u>_IrrGetMeshVertices</u>, <u>_IrrGetMeshIndices</u>, <u>_IrrAddMeshToScene</u>, <u>_CreateVertStruct</u>

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $orgMesh = IrrGetMesh( "./media/zumlin.md2" )
local $tVertex; variable for the vertex array struct
; copy the vertex information into the array:
local $vertex_count = _IrrGetMeshVertices( $orgMesh, 0, $tVertex)
local $i
for $i = 0 to $vertex count - 1; itterate through all of the vertices
  ; shrink vertex Y location by half its size, then change vertex colour value
  SetVertStruct($tVertex, $i, $VERT Y, GetVertStruct($tVertex, $i,
$VERT_Y) * 0.5)
    _SetVertStruct($tVertex, $i, $VERT_VCOLOR, _IrrMakeARGB(0,
Random(0,255), Random(0,255), Random(0,255)))
next; $i
; create a second mesh with the modified vertices data and unmodified
indices:
local $aIndices ; variable that will hold array of indices:
_IrrGetMeshIndices($orgMesh, 0, $aIndices)
local $secondMesh = IrrCreateMesh("secondMesh", $tVertex, $aIndices )
local $texture = _IrrGetTexture(".\media\default_texture.png")
```

```
; add both meshes and a camera to the scene:
local $nodeCube1 = IrrAddMeshToScene( $orgMesh )
_IrrSetNodeMaterialTexture( $nodeCube1,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeCube1, $IRR_EMF_LIGHTING, $IRR_OFF
)
local $nodeCube2 = _IrrAddMeshToScene( $secondMesh )
_IrrSetNodeMaterialFlag( $nodeCube2, $IRR_EMF_LIGHTING, $IRR_OFF
)
_IrrSetNodePosition($nodeCube2, 0, 0, 30)
_IrrAddCamera(50, 0, 30, 0, 0, 18)
WHILE _IrrRunning()
  _IrrBeginScene(0, 0, 25)
  _IrrDrawScene()
  _IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrAddHillPlaneMesh

Creates a hill plane mesh that represents a simple terrain.

#Include <au3Irrlicht2.au3>
_IrrAddHillPlaneMesh(\$s_Name, \$f_TileSizeX, \$f_TileSizeY,
\$i_TileCountX, \$i_TileCountY, \$h_Material = 0, \$f_HillHeight = 0,
\$f_CountHillsX = 0, \$f_CountHillsY = 0, \$f_TextureRepeatCountX = 1,
\$f_TextureRepeatCountY = 1)

Parameters

\$s_Name	
\$f_TileSizeX	
\$f_TileSizeY	
\$i_TileCountX	
\$i_TileCountY	
\$h_Material	
\$f_HillHeight	
\$f_CountHillsX	
\$f_CountHillsY	
<pre>\$f_TextureRepeatCountX</pre>	
\$f_TextureRepeatCountY	

Return Value

Success: Handle to a Terrain Mesh Failure: False and @error 1

Remarks

Many properties have default values allowing a mesh to be created with a simple call.

Related

IrrWriteMesh

Write the first frame of the supplied animated mesh out to a file using the specified file format.

#Include <au3Irrlicht2.au3>
_IrrWriteMesh(\$h_Mesh, \$i_FileFormat, \$s_Filename)

Parameters

\$h_Mesh	Handle to mesh object
	Format to write the file as: \$EMWT_IRR_MESH - Irrlicht Native mesh writer, for static .irrmesh files. \$EMWT_COLLADA - COLLADA mesh writer for .dae and .xml files. \$EMWT_STL - STL mesh writer for .stl files.
\$s_Filename	File name to save as.

Return Value

Success: True Failure: False and set @error @error 1 - Autolt DllCall failed. @error 2 - Could not get mesh writer object. @error 3 - Could not open file.

Remarks

None

Related

_IrrCreateMesh, _IrrGetMesh

Example

```
#include "au3Irrlicht2.au3"
```

Global \$hMD2Mesh Global \$hMeshTexture Global \$hSceneNode Global \$hCamera Global \$hIrrMesh Global \$sIrrMesh = "ZumlinStaticMesh.irrmesh"

_IrrStart()

```
$hMD2Mesh = IrrGetMesh("./media/zumlin.md2")
$hMeshTexture = IrrGetTexture("./media/zumlin.pcx")
$hSceneNode = IrrAddMeshToScene($hMD2Mesh)
_IrrSetNodeMaterialTexture($hSceneNode, $hMeshTexture, 0)
_IrrSetNodeMaterialFlag($hSceneNode, $IRR_EMF_LIGHTING,
$IRR OFF)
_IrrSetNodePosition($hSceneNode, 0, 0, 20)
If IrrWriteMesh($hMD2Mesh, $EMWT IRR MESH, $sIrrMesh) And
FileExists($sIrrMesh) Then
  $hIrrMesh = IrrGetMesh($sIrrMesh)
  $hSceneNode = _IrrAddMeshToScene($hIrrMesh)
  _IrrSetNodeMaterialTexture($hSceneNode, $hMeshTexture, 0)
  _IrrSetNodeMaterialFlag($hSceneNode, $IRR_EMF_LIGHTING,
$IRR OFF)
  IrrSetNodePosition($hSceneNode, 0, 0, -20)
  FileDelete($sIrrMesh)
EndIf
```

```
$hCamera = _IrrAddCamera(50, 0, 0, 0, 0, 0)
```

```
While _IrrRunning()
_IrrBeginScene(240, 255, 255)
_IrrDrawScene()
_IrrEndScene()
WEnd
```

_IrrStop()
Copy to Clipboard

IrrRemoveMesh

Removes a mesh from the scene cache, freeing up resources.

#Include <au3Irrlicht2.au3>
_IrrRemoveMesh(\$h_Mesh)

Parameters

\$h_Mesh | Handle of a mesh object

Return Value

Success: True Failure: False

Remarks

None

Related

<u>_IrrGetMesh</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $camera = _IrrAddCamera(2,2,2, 0,0,0 )
```

```
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $sceneNode = _IrrAddMeshToScene( $mesh )
```

_IrrRemoveMesh(\$mesh); it's in the scene, mesh no longer needed!

```
local $texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
```

```
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrClearUnusedMeshes

Clears all meshes that are held in the mesh cache but not used anywhere else.

#Include <au3Irrlicht2.au3>
_IrrClearUnusedMeshes()

Parameters

None.	

Return Value

Success: True Failure: False

Remarks

Any references to these meshes will become invalid.

Related

IrrSetMeshHardwareAccelerated

Set the supplied mesh as a Hardware Accelerated object.

#Include <au3Irrlicht2.au3>
_IrrSetMeshHardwareAccelerated(\$h_mesh, \$i_frame = 0)

Parameters

\$h_mesh	Handle of a mesh object	
\$i_frame	Frame number	

Return Value

Success: True Failure]: False

Remarks

This offloads the verticies and indicies to hardware support on the graphics card, making the process of rendering those meshes much faster.

The feature must be supported on the graphics card and the object must contain over 500 vertices for the operation to be successful.

This operation is applied to all mesh buffers in the mesh.

Related

IrrGetMeshIndexCount

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetMeshIndexCount(\$h_Mesh, \$i_Frame, \$i_MeshBuffer = 0)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetMeshIndices

Gets the list of indices in a mesh and copies them into the supplied variable.

```
#Include <au3Irrlicht2.au3>
_IrrGetMeshIndices($h_Mesh, $i_FrameNumber, ByRef $a_IndicesArray,
$i_MeshBuffer = 0)
```

Parameters

\$h_Mesh	Handle to a mesh object
\$i_FrameNumber	Frame number of the mesh to get indices from (should be 0 for static meshes).
\$i_MeshBuffer	[optional] Mesh buffer to access.

Return Value

Success: Number of indices returned in the array $a_IndicesArray$. Failure: False and @error = 1

Remarks

[todo]

Related

_IrrSetMeshIndices, _IrrGetMeshIndexCount, _IrrGetMeshVertices

IrrSetMeshIndices

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetMeshIndices(\$h_Mesh, \$i_FrameNumber, ByRef \$a_IndicesArray,
\$i_MeshBuffer = 0)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetMeshVertexCount

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetMeshVertexCount(\$h_Mesh, \$i_Frame, \$i_MeshBuffer = 0)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetMeshVertices

Gets the list of vertices in a mesh and copies them into the supplied variable.

```
#Include <au3Irrlicht2.au3>
_IrrGetMeshVertices($h_Mesh, $i_FrameNumber, ByRef $tVertex,
$i_MeshBuffer = 0)
```

Parameters

\$h_Mesh	Handle to a mesh object
\$i_FrameNumber	Frame number of the mesh to get vertices from (should be 0 for static meshes).
\$i_MeshBuffer	[optional] Mesh buffer to access.

Return Value

Success: Number of vertices returned in the vertex array struct t. Failure: False and @error = 1

Remarks

Each vertex represents a point in the mesh that is the corner of one of the group of triangles that is used to construct the mesh. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0. If the mesh contains a number of mesh buffers you can specific which mesh buffer you want to access, if you omit this parameter mesh buffer 0 will be used.

Related

<u>___SetVertStruct</u>, <u>__GetVertStruct</u>, <u>_IrrSetMeshVertices</u>, _<u>IrrGetMeshVertexCount</u>, <u>_IrrGetMeshIndices</u>

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $mesh = IrrGetMesh( "./media/cube.x" )
local $tVertex; variable for the vertex array struct
; copy the vertex information into the array
local $vertex count = IrrGetMeshVertices( $mesh, 0, $tVertex)
local $i
for $i = 0 to $vertex_count - 1; itterate through all of the vertices
  ; shrink vertex X location by half its size, then change vertex colour value
  __SetVertStruct($tVertex, $i, $VERT_X, __GetVertStruct($tVertex, $i,
$VERT X) * 0.5)
  SetVertStruct($tVertex, $i, $VERT VCOLOR, IrrMakeARGB(0,
Random(0,255), Random(0,255), Random(0,255)))
next; $i
; copy the altered vertex infomation back to the mesh
IrrSetMeshVertices( $mesh, 0, $tVertex )
; add mesh and camera to the scene:
local $nodeCube = _IrrAddMeshToScene( $mesh )
_IrrSetNodeMaterialFlag( $nodeCube, $IRR_EMF LIGHTING, $IRR OFF )
IrrSetNodePosition($nodeCube, -0.5, -0.5, 5)
_IrrAddFPSCamera($IRR_NO_OBJECT, 5, 0.01)
```

```
WHILE _IrrRunning()
_IrrBeginScene( 0, 0, 25 )
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrSetMeshVertices

This sets the value of the list of vertices in a mesh copying them from the supplied vertex array struct.

#Include <au3Irrlicht2.au3>
_IrrSetMeshVertices(\$h_Mesh, \$i_FrameNumber, ByRef \$tVertex,
\$i_MeshBuffer = 0)

Parameters

\$h_Mesh	Handle to a mesh object
\$i_FrameNumber	Frame number of the mesh to write vertices to (should be 0 for static meshes).
\$i_MeshBuffer	[optional] Mesh buffer to access.

Return Value

Success: None. Failure: False and @error = 1

Remarks

Each vertex represents a point in the mesh that is the corner of one of the group of triangles that is used to construct the mesh. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0. If the mesh contains a number of mesh buffers you can specific which mesh buffer you want to access, if you omit this parameter mesh buffer 0 will be used.

Related

<u>IrrGetMeshVertices</u>, <u>CreateVertStruct</u>

Example

```
#include "au3Irrlicht2.au3"
```

_IrrStart()

```
local $mesh = _IrrGetMesh( "./media/cube.x" )
```

local \$tVertex; variable for the vertex array struct
; copy the vertex information into the array
local \$vertex_count = _IrrGetMeshVertices(\$mesh, 0, \$tVertex)

local \$i

```
for $i = 0 to $vertex_count - 1 ; itterate through all of the vertices
  ; shrink vertex X location by half its size, then change vertex colour value
```

__SetVertStruct(\$tVertex, \$i, \$VERT_X, __GetVertStruct(\$tVertex, \$i, \$VERT_X) * 0.5)

```
___SetVertStruct($tVertex, $i, $VERT_VCOLOR, _IrrMakeARGB(0, Random(0,255), Random(0,255), Nandom(0,255) ) ) next ; $i
```

```
; copy the altered vertex infomation back to the mesh
_IrrSetMeshVertices( $mesh, 0, $tVertex )
```

```
; add mesh and camera to the scene:
local $nodeCube = _IrrAddMeshToScene( $mesh )
_IrrSetNodeMaterialFlag( $nodeCube, $IRR_EMF_LIGHTING, $IRR_OFF )
_IrrSetNodePosition($nodeCube, -0.5, -0.5, 5)
```

```
_IrrAddFPSCamera($IRR_NO_OBJECT, 5, 0.01)
```

```
WHILE _IrrRunning()
```

```
_IrrBeginScene( 0, 0, 25 )
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrAddMeshToScene

Adds a mesh to the scene as a new 3D node.

#Include <au3Irrlicht2.au3>
_IrrAddMeshToScene(\$h_Mesh)

Parameters

\$h_Mesh Handle of a mesh object

Return Value

Success: Handle of the new node in the scene Failure: False

Remarks

None

Related

_IrrGetMesh

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $camera = _IrrAddCamera(2,2,2, 0,0,0 )
local $mesh = _IrrGetMesh( ".\media\capsuleX.obj" )
local $sceneNode = _IrrAddMeshToScene( $mesh )
local $texture = _IrrGetTexture(".\media\default_texture.png")
_IrrSetNodeMaterialTexture( $sceneNode, $texture, 0)
_IrrSetNodeMaterialFlag( $sceneNode, $IRR_EMF_LIGHTING, $IRR_OFF
)
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrAddMeshToSceneAsOcttree

Adds a mesh to the scene as a new 3D node, optimised with an Octtree.

#Include <au3Irrlicht2.au3>
_IrrAddMeshToSceneAsOcttree(\$h_Mesh)

Parameters

\$h_Mesh Handle of an mesh object

Return Value

Success: Handle to the irrlicht node object Failure: False

Remarks

This method optimise's the mesh with an Octtree, this is particularly useful for maps where there is a lot of geometry in the mesh but little of it can be seen at any one time.

Optimizing your node with this function will result in a large increase in performance.

Related

None.

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
_IrrAddZipFile( "./media/map-20kdm2.pk3", $IRR_IGNORE_CASE,

$IRR_IGNORE_PATHS )

$BSPMesh = _IrrGetMesh( "20kdm2.bsp" )

local $BSPNode = _IrrAddMeshToSceneAsOcttree( $BSPMesh )
```

```
local $camera = _IrrAddFPSCamera()
_IrrSetNodePosition( $camera, 1750, 149, 1369 )
```

```
WHILE _IrrRunning()
_IrrBeginScene(50, 50, 50)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrAddStaticMeshForNormalMappingToScene

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddStaticMeshForNormalMappingToScene(\$h_Mesh)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrLoadScene

[todo]

#Include <au3Irrlicht2.au3>
_IrrLoadScene(\$s_Filename)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSaveScene

[todo]

#Include <au3Irrlicht2.au3>
_IrrSaveScene(\$s_Filename)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetSceneNodeFromId

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetSceneNodeFromId(\$i_ID)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetSceneNodeFromName

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetSceneNodeFromName(\$s_Name)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddBillBoardToScene

Adds a billboard to the scene of the specified size and at the specified position.

#Include <au3Irrlicht2.au3>
_IrrAddBillBoardToScene(\$f_XSize, \$f_YSize, \$f_XPos = 0.0, \$f_YPos =
0.0, \$f_ZPos = 0.0)

Parameters

\$f_XSize	X size of the node
\$f_YSize	Y size of the node
\$f_XPos	[optional] X position
\$f_YPos	[optional] Y position
\$f_ZPos	[optional] Z position

Return Value

Success: Handle of the new billboard scene node Failure: False

Remarks

A billboard is a flat 3D textured sprite that always faces towards the camera. You need to texture this element with a separate command.

Related

_IrrSetNodeMaterialTexture, _IrrSetNodeMaterialFlag

Example

```
#include "au3Irrlicht2.au3"
```

_IrrStart()

```
local $BillboardTexture = _IrrGetTexture( "./media/au3irr2_logo.jpg" )
local $Billboard = _IrrAddBillBoardToScene( 200.0,102, 0.0,0.0,100.0 )
```

_IrrSetNodeMaterialTexture(\$Billboard, \$BillboardTexture, 0)
_IrrSetNodeMaterialFlag(\$Billboard, \$IRR_EMF_LIGHTING, \$IRR_OFF)

local \$Camera = _IrrAddFPSCamera()

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrSetBillBoardSize

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetBillBoardSize(\$h_Node, \$f_Width, \$f_Height)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddBillboardTextSceneNode

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddBillboardTextSceneNode(\$h_Font, \$s_Text, \$f_XSize, \$f_YSize,
\$f_XPos=0, \$f_YPos=0, \$f_ZPos=0, \$h_Parent=0,
\$i_TopRGBA=0xFFFFFFF, \$i_BottomRGBA=0xFFFFFFFF)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

[todo: functionName, functionName]

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $font = _IrrGetFont( "./media/fonthaettenschweiler.bmp" )
```

```
_IrrAddBillboardTextSceneNode( $font, "au3Irrlicht2 ...", 30.0, 15.0, 0.0, 0.0, 20.0, 0, _
```

```
_IrrMakeARGB(0, 255, 0, 0), _IrrMakeARGB(0, 255,
```

255, 0))

```
_IrrAddBillboardTextSceneNode( $font, "... what else?", 30.0, 15.0, 0.0, -5.0, 17.0, 0, _
```

```
_IrrMakeARGB(0, 255, 255, 0), _IrrMakeARGB(0, 0,
```

255, 0))

```
_IrrAddFPSCamera()
```

```
WHILE _IrrRunning()
_IrrBeginScene(0,0,25)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

```
Copy to Clipboard
```

IrrAddParticleSystemToScene

Adds a particle system to the irrlicht scene manager.

#Include <au3Irrlicht2.au3>
_IrrAddParticleSystemToScene(\$b_AddEmitter, \$h_Parent = 0, \$i_Id = -1,
\$f_PosX = 0, \$f_PosY = 0, \$f_PosZ = 0, \$f_RotX = 0, \$f_RotY = 0,
\$f_RotZ = 0, \$f_ScaleX = 1, \$f_ScaleY = 1, \$f_ScaleZ = 1)

Parameters

\$b_AddEmitter	Whether default emitter shall be created or not: \$IRR_NO_EMITTER - For no default emitter (this is probably the option you will use and you will then add a specific emitter later). IRR_DEFAULT_EMITTER - To create a default emitter that ejects a thin vertical stream of particles.
\$h_Parent	[optional] Handle of scene node the particle shall be attached to (0 means attach to the root scene node)
\$i_Id	[optional] Assigns given integer as ID to the created particle system.
\$f_PosX, \$f_PosY, \$f_PosZ	[optional] Set position of particle system in the Irrlicht scene.
\$f_RotX, \$f_RotY, \$f_RotZ	[optional] Rotate the particle system along x, y, z axes (0-360).
\$f_ScaleX, \$f_ScaleY, \$f_ScaleZ	[optional] Scaling factors for created particle system.

Return Value

success: Handle of the created particle system. failure: False

Remarks

A particle system is an object that creates and manages hundreds of small billboard like objects that are used to represent smoke, rain and other natural effects.

Once created you then need to add emitters and affectors to create and control the particles.

Related

```
<u>_IrrAddParticleEmitter</u>, <u>_IrrAddFadeOutParticleAffector</u>,
<u>_IrrAddGravityParticleAffector</u>, <u>_IrrAddParticleAttractionAffector</u>,
<u>_IrrAddRotationAffector</u>
```

Example

#include "au3Irrlicht2.au3"

_IrrStart()

; add particle system with default emitter to the irrlicht scene manager and scale it for more depth

local \$particleSystem =

_IrrAddParticleSystemToScene(\$IRR_DEFAULT_EMITTER, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 3, 6)

; load a grey smoke like image for the particle

local \$ParticleTexture = _IrrGetTexture("./media/ParticleGrey.bmp")

; apply the texture to the particles system to be drawn across each particles surface

_IrrSetNodeMaterialTexture(\$particleSystem, \$ParticleTexture, 0) ; particle system is not affected by lighting so make it self illuminating

_IrrSetNodeMaterialFlag(\$particleSystem, \$IRR_EMF_LIGHTING, \$IRR_OFF) ; don't draw black parts of the particle texture: _IrrSetNodeMaterialType (\$particleSystem, \$IRR_EMT_TRANSPARENT_ADD_COLOR)

```
local $nodeCamera = _IrrAddCamera( 80,0,0, 20,40,0 )
```

```
WHILE _IrrRunning()
_IrrBeginScene( 0,0,50 )
_IrrDrawScene()
_IrrEndScene()
```

WEND

_IrrStop()

```
Copy to Clipboard
```

IrrAddSkyBoxToScene

Adds a skybox node to the scene.

#Include <au3Irrlicht2.au3>
_IrrAddSkyBoxToScene(\$h_UpTexture, \$h_DownTexture, \$h_LeftTexture,
\$h_RightTexture, \$h_FrontTexture, \$h_BackTexture)

Parameters

\$h_UpTexture	Handle of the top side texture
<pre>\$h_DownTexture</pre>	Handle of the bottom side texture
\$h_LeftTexture	Handle of the left side texture
\$h_RightTexture	Handle of the right side texture
\$h_FrontTexture	Handle of the front side texture
\$h_BackTexture	Handle of the back side texture

Return Value

success: Handle of the skybox node failure: False

Remarks

A skybox is a huge hollow cube that encapsulates the entire scene and has a different texture applied to each of its six surfaces to represent a distant sky or matte scene.

Related

_IrrAddSkyDomeToScene

Example

```
#include <au3Irrlicht2.au3>
_IrrStart()
local $nodeSkyBox = _IrrAddSkyBoxToScene( _
    _IrrGetTexture("./media/irrlicht2_up.jpg"), _
    _IrrGetTexture("./media/irrlicht2_dn.jpg"), _
    _IrrGetTexture("./media/irrlicht2_rt.jpg"), _
    _IrrGetTexture("./media/irrlicht2_lf.jpg"), _
    _IrrGetTexture("./media/irrlicht2_ft.jpg"), _
    _IrrGetTexture("./media/irrlicht2_bk.jpg"))
local $nodeCamera = _IrrAddFPSCamera()
WHILE _IrrRunning()
  _IrrBeginScene( 240, 255, 255 )
  _IrrDrawScene()
  _IrrEndScene()
WEND
_IrrStop()
 Copy to Clipboard
```

IrrAddSkyDomeToScene

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddSkyDomeToScene(\$h_Texture, \$i_HorizontalRes, \$i_VerticalRes,
\$d_TexturePercent, \$d_SpherePercent, \$d_SphereRadius = 1000.0)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

<u>_IrrAddSkyBoxToScene</u>, <u>_IrrSetSkyDomeColor</u>, <u>_IrrSetSkyDomeColorBand</u>, <u>_IrrSetSkyDomeColorPoint</u>

IrrAddTestSceneNode

Adds a simple cube node to the scene

#Include <au3Irrlicht2.au3>
_IrrAddTestSceneNode()

Parameters

None.

Return Value

success: Handle of the cube scene node failure: False

Remarks

The test scene node is a cube with fixed dimensions mainly for test purposes.

Related

_IrrAddCubeSceneNode, _IrrAddSphereSceneNode

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCamera = _IrrAddCamera(10,10,10, 0,0,0 )
local $nodeTest = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrAddCubeSceneNode

Adds a cube object to the scene with the specified dimensions.

#Include <au3Irrlicht2.au3>
_IrrAddCubeSceneNode(\$f_Size)

Parameters

\$f_Size Edge length of the cube.

Return Value

success: Handle of the cube scene node failure: False

Remarks

None

Related

_IrrAddTestSceneNode, _IrrAddSphereSceneNode

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCamera = _IrrAddCamera(10,10,10, 0,0,0 )
local $nodeTest = _IrrAddCubeSceneNode(10)
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrAddSphereSceneNode

Adds a simple sphere object to the scene

#Include <au3Irrlicht2.au3>
_IrrAddSphereSceneNode(\$f_Size, \$i_PolyCount = 16)

Parameters

\$f_Size	Radius of the sphere
\$i_PolyCount	[optional] Level of detail for the sphere. Too high values could produce a very high density mesh and affect your frame rate adversely.

Return Value

success: Handle of the sphere scene node failure: False

Remarks

None

Related

_IrrAddTestSceneNode, _IrrAddCubeSceneNode

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCamera = _IrrAddCamera(10,10,10,0,0,0)
local $nodeTest = _IrrAddSphereSceneNode(8,32)
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrAddWaterSurfaceSceneNode

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddWaterSurfaceSceneNode(\$h_Mesh, \$f_WaveHeight = 2.0,
\$f_WaveSpeed = 300.0, \$f_WaveLength = 10.0, \$h_Parent = 0, \$i_ID = -1,
\$f_PosX = 0, \$f_PosY = 0, \$f_PosZ = 0, \$f_RotX = 0, \$f_RotY = 0,
\$f_RotZ = 0, \$f_ScaleX = 1.0, \$f_ScaleY = 1.0, \$f_ScaleZ = 1.0)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddZoneManager

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddZoneManager(\$f_NearDistance=0, \$f_FarDistance=12000)

Parameters

Inaramii	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddClouds

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddClouds(\$h_Texture, \$i_Lod, \$i_Depth, \$i_Density)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

[todo: functionName, functionName]

Example

#include "au3Irrlicht2.au3"

Global \$hCloudNode, \$hCloudTexture

_IrrStart()

```
$hCloudTexture = _IrrGetTexture("./media/cloud4.png")
$hCloudNode = _IrrAddClouds($hCloudTexture, 10, 1, 200)
_IrrSetNodePosition($hCloudNode, 0, 200, 0)
```

_IrrAddCamera(50, 0, 0, 0, 0, 0)

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(128, 128, 255)
_IrrDrawScene()
_IrrEndScene()
WEnd
```

```
_IrrStop()
Copy to Clipboard
```

IrrAddLensFlare

Adds a lens flare patch object to the scene, this object simulates the effect of bright lights on the optics of a camera.

#Include <au3Irrlicht2.au3>
_IrrAddLensFlare(\$h_Texture)

Parameters

h_Texture Handle to texture (image containing a series of 128x128 images representing stages of the the lens flare).

Return Value

Success: Handle to a flare node Failure: False and @error 1

Remarks

[todo]

Related

_IrrSetFlareScale, _IrrGetTexture

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $hFlare = _IrrAddLensFlare(_IrrGetTexture( "./media/flares.jpg" ))
_IrrSetNodePosition($hFlare, 300,100,1000 )
```

```
_IrrAddFPSCamera()
_IrrHideMouse()
```

```
While _IrrRunning()
_IrrBeginScene( 180, 225, 255 )
_IrrDrawScene()
_IrrEndScene()
```

WEnd

```
_IrrStop()
```

Copy to Clipboard

IrrAddGrass

Adds a grass object to the scene.

#Include <au3Irrlicht2.au3>
_IrrAddGrass(\$h_Terrain, \$i_X, \$i_Y, \$i_PatchSize, \$f_FadeDistance,
\$i_Crossed, \$f_GrassScale, \$i_MaxDensity, \$i_DataPosX, \$i_DataPosY,
\$h_HeightMap, \$h_TextureMap, \$h_GrassMap, \$h_GrassTexture)

Parameters

\$h_Terrain	[explanation]
\$i_X	X Positon of grass patch.
\$i_Y	Y Positon of grass patch.
\$i_PatchSize	Size of grass patch
\$f_FadeDistance	Distance at which the number of displayed grass elements in that patch are reduced. If this is set to 1.0 then when the cameral is inside the patch all of grass will be displayed but once outside less and less will be shown. By increasing this to 2.0 then all of the grass is shown until the camera is two patches distant. This gives a better appearence but reduces performance as more grass has to be drawn.
\$i_Crossed	 IRR_ON or IRR_OFF. When off each piece of grass is a separate entity with its own position and rotation. When On grass is paired up and placed to form a cross. Crossed grass can have a better appearance as you rotate around it. However individual grass can give the impression that there is more of it and you can therefore reduce the number of grass blades and increase performance.

\$f_GrassScale	Scale of a grass patch.
\$i_MaxDensity	number of individual clumps of folliage that are created.
\$i_DataPosX	X position of large bitmap associated with a tiled terrain and allow the color information to be taken from an offset position on the bitmap.
\$i_DataPosY	Y position of large bitmap associated with a tiled terrain and allow the color information to be taken from an offset position on the bitmap.
\$h_TextureMap	Handle to an image that contains the height of the terrain onto which the grass is placed.
\$h_GrassMap	Handle to an image used to adjust the height and density of the grass. For example you might have a patch where you dont want to see any grass or a barren patch where you want short stubble.
\$h_GrassTexture	Handle to a Texture used for the grass. This RGBA image is automatically broken up into a number of sections that are used to texure different clumps of grass.

Return Value

Success: Handle to a Grass node. Failure: False and @error 1

Remarks

Grass objects are associated with terrain and tile terrain objects and are used to place small billboard objects into the scene representing folliage, this implementation of grass creates a large number of grass objects already positioned across the terrain and then dynamically shows or hides them depending on where the camera is within the scene. The grass is also affected with a wind modifier that gently moves the grass as if it were caught in the wind.

By setting the speed of the wind to zero the grass will become static and you will see an increase in performance.

Grass usually looks best when it is closely matched to the color of the terrain and to assist with this a new Material Type has been added IRR_EMT_TRANSPARENT_ADD_ALPHA_CHANNEL_REF that adds the color of grass texture to the color of the grass which is automatically set to the color of the terrain that it lies upon.

Related

_IrrGetGrassDrawCount, _IrrSetGrassDensity, _IrrSetGrassWind

IrrSetShadowColor

Sets the colour of shadows cast by objects in the scene.

#Include <au3Irrlicht2.au3>
_IrrSetShadowColor(\$i_Alpha, \$i_Red, \$i_Green, \$i_Blue)

Parameters

\$i_Alpha	Alpha blend for the shadow. Value of 128 would mean a half washed out shadow which gives the appearence of ambient light in the room illuminating the shadowed surface.
\$i_Red, \$i_Green, \$i_Blue	Colour values for shadow colour from 0 to 255.

Return Value

Success: True Failure: False

Remarks

The shadow colour is a global property for the whole scene (however you can change it when moving into different areas of your scene). If you are observing a bright scene you might use a light grey shadow instead of a heavy black shadow to add to realism.

Related

<u>_IrrAddNodeShadow</u>, <u>_IrrAddLight</u>

Example

#include "au3Irrlicht2.au3"

```
_IrrStart( $IRR_EDT_OPENGL, 800, 600, $IRR_BITS_PER_PIXEL_32, _
$IRR_WINDOWED, $IRR_SHADOWS, $IRR_IGNORE_EVENTS,
$IRR_VERTICAL_SYNC_ON )
```

```
local $texture = _IrrGetTexture(".\media\au3irr2_logo.jpg")
```

```
local $meshReceive = _IrrAddHillPlaneMesh( "Plane", 8, 8, 1, 1)
local $nodeReceive = _IrrAddMeshToScene( $meshReceive )
_IrrSetNodeMaterialTexture( $nodeReceive, $texture, 0)
```

```
local $meshCast = _IrrGetMesh(".\media\sphere.obj")
local $nodeCast = _IrrAddMeshToScene($meshCast)
_IrrSetNodeMaterialTexture( $nodeCast, $texture, 0)
_IrrAddFlyCircleAnimator($nodeCast, 1, 3, 1, 3, 0.0005)
```

```
local $nodeLight = _IrrAddLight($IRR_NO_PARENT, 0,50,0, 1,0.1,0.1,
1000.0)
_IrrAddNodeShadow($nodeCast)
_IrrSetShadowColor(128, 255, 0, 0)
```

```
local $nodeCamera = _IrrAddCamera(-5,5,-6, 0,0,0 )
```

```
WHILE _IrrRunning()
_IrrBeginScene(100, 100, 100)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop() Copy to Clipboard

_IrrSetFog

Set the properties of fog in the scene

#Include <au3Irrlicht2.au3>
_IrrSetFog(\$i_Red, \$i_Green, \$i_Blue, \$i_FogType, \$f_FogStart,
\$f_FogEnd, \$f_Density = 0.025)

Parameters

\$i_Red,	Define colour of the fog (0-255).
\$i_Green,	Should be set to the same colour as the scene sky so the
\$i_Blue	scene fogs out nicely into nothing.
\$i_FogType	Defines how the fog is calculated: \$IRR_LINEAR_FOG - computed as [end - distance / end - start], density value is not used. \$IRR_EXPONENTIAL_FOG - computed as [1 / (2.718^(distance * densitiy))], both start and end values are not used.
\$f_FogStart,	Distances at which the fog starts and at which it reaches its maximum density.
\$f_FogEnd	Values are ignored for exponential fog.
\$f_Density	[optional] Determines how quickly the exponential change takes place, with value from 0 to 1. Example: A value of 0.025 equals 20% visibility at 50 units distance. Value is ignored for linear fog.

Return Value

success: True failure: False

Remarks

None.

Related

None.

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $nodeCamera = _IrrAddCamera(10, 10, 10, 0, 0, 0)
local $nodeCube = _IrrAddCubeSceneNode(12)
_IrrSetNodeMaterialTexture( $nodeCube,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeCube, $IRR_EMF_LIGHTING, $IRR_OFF )
_IrrSetNodeMaterialFlag( $nodeCube, $IRR_EMF_FOG_ENABLE,
$IRR_ON)
IrrSetFog (100,100,100, $IRR EXPONENTIAL FOG, 0.0, 0.0, 0.15)
WHILE _IrrRunning()
  _IrrBeginScene(100, 100, 100)
  _IrrDrawScene()
  _IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrDraw3DLine

Draws a line onto the display using 3D co-ordinates and a specified colour.

#Include <au3Irrlicht2.au3>
_IrrDraw3DLine(\$f_XStart, \$f_YStart, \$f_ZStart, \$f_XEnd, \$f_YEnd,
\$f_ZEnd, \$i_Red, \$i_Green, \$i_Blue)

Parameters

\$f_XStart, \$f_YStart, \$f_ZStart	Defines start point for the 3D-line.
\$f_XEnd, \$f_YEnd, \$f_ZEnd	Defines end point for the 3D-line.
\$i_Red, \$i_Green, \$i_Blue	Colour values for the 3D-Line (0-255).

Return Value

success: True failure: False

Remarks

The lines are not part of the Irrlicht scene but drawn before and separately. They need to be redrawn for every new frame.

Related

_IrrBeginScene, _IrrEndScene()

Example

```
#include "au3Irrlicht2.au3"
_IrrStart()
local $i
local $nodeCamera = _IrrAddFPSCamera()
_IrrSetCameraTarget($nodeCamera, 50, 0, 50)
WHILE _IrrRunning()
  _IrrBeginScene(0, 0, 50)
  ; draw a grid to sreen before drawing the scene:
  for $i = 0 to 250 step 25
       _IrrDraw3DLine($i, -25, 0, $i, -25, 250, 255, 255, 0)
       _IrrDraw3DLine(0, -25, $i, 250, -25, $i, 255, 255, 0)
  next; $i
  _IrrDrawScene()
  _IrrEndScene()
WEND
_IrrStop()
Copy to Clipboard
```

IrrSetSkyDomeColor

Set the color of the verticies in the skydome.

#Include <au3Irrlicht2.au3>
_IrrSetSkyDomeColor(\$h_Dome, \$i_HorizontalRed, \$i_HorizontalGreen,
\$i_HorizontalBlue, \$i_ZenithRed, \$i_ZenithGreen, \$i_ZenithBlue)

Parameters

\$h_Dome	[explanation]
\$i_HorizontalRed	Red color 0 - 255
\$i_HorizontalGreen	Green color 0 - 255
\$i_HorizontalBlue	Blue color 0 - 255
\$i_ZenithRed	Red color 0 - 255
\$i_ZenithGreen	Green color 0 - 255
\$i_ZenithBlue	Blue color 0 - 255

Return Value

Success: True Failure: False

Remarks

Two colors are defined one for the horizon and another for the top of the sky dome, this simulates the type of coloring effects you see in the sky. If you are using a full spherical skydome the horizontal color will be the color at the bottom of the skydome.

Related

<u>_IrrSetSkyDomeColorPoint</u>, <u>_IrrSetSkyDomeColorBand</u>, <u>_IrrSAddSkyDome</u>

IrrSetSkyDomeColorBand

Creates a horizontal band of color in the skydome.

#Include <au3Irrlicht2.au3>

- _IrrSetSkyDomeColorBand(\$h_Dome, \$i_HorizontalRed,
- \$i_HorizontalGreen, \$i_HorizontalBlue, \$i_BandVerticalPosition,
- \$f_BandFade, \$i_Additive)

Parameters

\$h_Dome	[explanation]
\$i_HorizontalRed	Red color 0 - 255
\$i_HorizontalGreen	Green color 0 - 255
\$i_HorizontalBlue	Blue color 0 - 255
\$i_BandVerticalPosition	Vertex at which you wish to create the band.
\$f_BandFade	Amount that the band is faded into the existing skydome color.
\$i_Additive	IRR_ON to add the color of the band to the existing color of the skydome or IRR_OFF to replace it.

Return Value

Success: True Failure: False

Remarks

This is mainly useful for creating additional bands of color at the horizon, where your sky is a graduation of blues and then in the morning you have a brighter golden band as the sun rises.

Related

_IrrSetSkyDomeColorPoint, _IrrSetSkyDomeColor, _IrrSAddSkyDome

IrrSetSkyDomeColorPoint

Set the color of the verticies in the skydome radiating out from a point.

#Include <au3Irrlicht2.au3>
_IrrSetSkyDomeColorPoint(\$h_Dome, \$i_Red, \$i_Green, \$i_Blue,
\$f_PosX, \$f_PosY, \$f_PosZ, \$f_Radius, \$f_PointFade, \$i_Additive)

Parameters

\$h_Dome	Handle to a SkyDome node.
\$i_Red	Red color 0 - 255
\$i_Green	Green color 0 - 255
\$i_Blue	Blue color 0 - 255
\$f_PosX	X position.
\$f_PosY	Y position.
\$f_PosZ	Z position.
\$f_Radius	Limit the distance of the coloring
\$f_PointFade	Amount that the band is faded into the existing skydome color.
\$i_Additive	IRR_ON to add the color of the band to the existing color of the skydome or IRR_OFF to replace it.

Return Value

Success: True Failure: False

Remarks

This is powerful effect that can be used to color parts of the skydome and

create effects to represent the glows of the rising sun or the moon in the sky.

Related

_IrrSetSkyDomeColorBand, _IrrSetSkyDomeColor, _IrrSAddSkyDome

IrrSetZoneManagerProperties

Sets the draw distances of nodes in the zone/distance management node and whether or not the zone manager is to accumulate the bounding boxes of its children as they are added.

#Include <au3Irrlicht2.au3>
_IrrSetZoneManagerProperties(\$h_ZoneManager, \$f_NearDistance,
\$f_FarDistance, \$i_AccumulateBoxes)

Parameters

\$h_ZoneManager	Handle to the zone node.
\$f_NearDistance	
\$f_FarDistance	
\$i_AccumulateBoxes	

Return Value

Success: True Failure: False

Remarks

[todo]

Related

[todo: functionName, functionName]

IrrSetZoneManagerBoundingBox

Allows the user to manually set the bounding box of a zone manager node.

#Include <au3Irrlicht2.au3>
_IrrSetZoneManagerBoundingBox(\$h_ZoneManager, \$f_X, \$f_Y, \$f_Z,
\$f_BoxWidth, \$f_BoxHeight, \$f_BoxDepth)

Parameters

\$h_ZoneManager	Handle to the zone node.
\$f_X	X position.
\$f_Y	Y position.
\$f_Z	Z position.
\$f_BoxWidth	Width of zone box.
\$f_BoxHeight	Height of zone box.
\$f_BoxDepth	Depth of zone box.

Return Value

Success: True Failure: False

Remarks

None.

Related

[todo: functionName, functionName]

IrrSetGrassDensity

Set the grass density.

#Include <au3Irrlicht2.au3>
_IrrSetGrassDensity(\$h_Grass, \$f_Density, \$f_Distance)

Parameters

\$h_Grass	Handle to the grass node.
\$f_Density	Number of grass nodes visible in the scene.
\$f_Distance	Distance at which they can be seen.

Return Value

Success: True Failure: False

Remarks

None.

Related

_IrrAddGrass, _IrrGetGrassDrawCount, _IrrSetGrassWind

IrrSetGrassWind

Set the grass wind effect.

#Include <au3Irrlicht2.au3>
_IrrSetGrassWind(\$h_Grass, \$f_Strength, \$f_Resolution)

Parameters

\$h_Grass	Handle to the grass node.
\$f_Strength	Strength of the wind.
\$f_Resolution	How often the effect is calculated.

Return Value

Success: True Failure: False

Remarks

By setting the resolution to zero the wind effect will be stopped and there will be a performance increase, however the wind effect adds significantly to the subtle atmosphere of the

however the wind effect adds significantly to the subtle atmosphere of the scene.

Related

_IrrAddGrass, _IrrGetGrassDrawCount, _IrrSetGrassDensity

IrrCreateBatchingMesh

Create a batching mesh that will be a collection of other meshes into a single source mesh.

#Include <au3Irrlicht2.au3>
_IrrCreateBatchingMesh()

Parameters

None.

Return Value

Success: Handle to a bactching mesh. Failure: False and @error 1

Remarks

The function of the batching mesh is to avoid the use of large numbers of nodes that adds an overhead to the rendering process that can significantly slow it down.

Where you have a forest with a thousand trees you will see a significant increase in performance by batching all of those trees into a smaller number of node.

While this is handled as an irr_mesh it should only be used with batching mesh commands.

Related

[todo: functionName, functionName]

See Also

_IrrAddToBatchingMesh, _IrrFinalizeBatchingMesh

IrrAddToBatchingMesh

Adds a mesh to the batching mesh at the specified position, rotation and scale.

#Include <au3Irrlicht2.au3>
_IrrAddToBatchingMesh(\$h_meshBatch, \$h_mesh, \$f_posX = 0.0, \$f_posY
= 0.0, \$f_posZ = 0.0, \$f_rotX = 0.0, \$f_rotY = 0.0, \$f_rotZ = 0.0, \$f_scaleX
= 1.0, \$f_scaleY = 1.0, \$f_scaleZ = 1.0)

Parameters

\$h_meshBatch	Handle to a bactching mesh as returned byIrrCreateBatchingMesh.
\$h_mesh	Handle to the meash to add.
\$f_posX	X position
\$f_posY	Y position
\$f_posZ	Z position
\$f_rotX	X rotation
\$f_rotY	Y rotation
\$f_rotZ	Z rotation
\$f_scaleX	X scale
\$f_scaleY	Y scale
\$f_scaleZ	Z scale

Return Value

Success: True Failure: False

Remarks

If each of your meshes requires a different texture you should call IrrSetMeshMaterialTexture for the mesh you are about to add prior to adding the mesh to the batch.

Related

_IrrCreateBatchingMesh, _IrrFinalizeBatchingMesh

IrrFinalizeBatchingMesh

Finalises the batching mesh.

#Include <au3Irrlicht2.au3>
_IrrFinalizeBatchingMesh(\$h_meshBatch)

Parameters

h_meshBatch Handle to a bactching mesh as returned by ______IrrCreateBatchingMesh.

Return Value

Success: Handle to a new mesh Failure: False and @error 1

Remarks

This should be called once all of the meshes have been added to the batching mesh.

The function returns a new mesh object that can be used in all standard mesh calls..

Related

_IrrCreateBatchingMesh, _IrrAddToBatchingMesh

IrrSetMeshMaterialTexture

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetMeshMaterialTexture(\$h_mesh, \$h_texture, \$i_index, \$i_buffer = 0)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

[todo: functionName, functionName]

IrrScaleMesh

[todo]

#Include <au3Irrlicht2.au3>
_IrrScaleMesh(\$h_mesh, \$f_scale, \$i_frame = 0, \$i_meshBuffer = 0,
\$h_sourceMesh = 0)

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

[todo: functionName, functionName]

IrrAddBeamSceneNode

Add a beam scene node.

#Include <au3Irrlicht2.au3>
_IrrAddBeamSceneNode()

Parameters

None.

Return Value

Success: Handle to a beam node. Failure: False and @error 1

Remarks

The beam is a special scene node that can be used to replicate beam effects like lasers and tracer gun fire.

This command simply adds the beam you should then make calls to set the beams properties.

Related

<u>IrrSetBeamSize</u>, <u>IrrSetBeamPosition</u>

Example

#include "au3Irrlicht2.au3"

```
Global $hBeamNode
Global $hTexture
Global $hCamera
```

_IrrStart()

```
$hBeamNode = _IrrAddBeamSceneNode()
_IrrSetBeamPosition($hBeamNode, 15.0, 20.0, 20.0, -15.0, -20.0, -20.0)
_IrrSetBeamSize($hBeamNode, 5.0)
$hTexture = _IrrGetTexture("./media/beam.png")
_IrrSetNodeMaterialTexture($hBeamNode, $hTexture, 0)
_IrrSetNodeMaterialType($hBeamNode,
$IRR_EMT_TRANSPARENT_ALPHA_CHANNEL)
```

\$hCamera = _IrrAddCamera(50, 0, 0, 0, 0, 0)

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEnd
```

_IrrStop() Copy to Clipboard

IrrSetBeamSize

Sets the width of a beam node.

#Include <au3Irrlicht2.au3>
_IrrSetBeamSize(\$h_BeamNode, \$f_Size)

Parameters

\$h_BeamNode	Handle to a beam node.
\$f_Size	Width of the beam node.

Return Value

Success: True Failure: False

Remarks

None.

Related

_IrrAddBeamSceneNode, _IrrSetBeamPosition

Example

#include "au3Irrlicht2.au3"

Global \$hBeamNode

Global \$hTexture Global \$hCamera

_IrrStart()

```
$hBeamNode = _IrrAddBeamSceneNode()
_IrrSetBeamPosition($hBeamNode, 15.0, 20.0, 20.0, -15.0, -20.0, -20.0)
_IrrSetBeamSize($hBeamNode, 5.0)
$hTexture = _IrrGetTexture("./media/beam.png")
_IrrSetNodeMaterialTexture($hBeamNode, $hTexture, 0)
_IrrSetNodeMaterialType($hBeamNode,
$IRR_EMT_TRANSPARENT_ALPHA_CHANNEL)
$hCamera = _IrrAddCamera(50, 0, 0, 0, 0, 0)
While _IrrRunning() And Sleep(10)
_IrrBeginScene(0, 0, 0)
_IrrEndScene()
WEnd
_IrrStop()
```

Copy to Clipboard

IrrSetBeamPosition

Sets the start and end positions of a beam node. The beam will stretch between the two nodes.

#Include <au3Irrlicht2.au3>
_IrrSetBeamPosition(\$h_BeamNode, \$f_SX, \$f_SY, \$f_SZ, \$f_EX, \$f_EY,
\$f_EZ)

Parameters

\$h_BeamNode	Handle to a beam node.
\$f_SX	X start position
\$f_SY	Y start position
\$f_SZ	Z start position
\$f_EX	X end position
\$f_EY	X end position
\$f_EZ	X end position

Return Value

Success: True Failure: False

Remarks

None.

Related

_IrrAddBeamSceneNode, _IrrSetBeamSize

Example

```
#include "au3Irrlicht2.au3"
```

Global \$hBeamNode Global \$hTexture Global \$hCamera

_IrrStart()

```
$hBeamNode = _IrrAddBeamSceneNode()
_IrrSetBeamPosition($hBeamNode, 15.0, 20.0, 20.0, -15.0, -20.0, -20.0)
_IrrSetBeamSize($hBeamNode, 5.0)
$hTexture = _IrrGetTexture("./media/beam.png")
_IrrSetNodeMaterialTexture($hBeamNode, $hTexture, 0)
_IrrSetNodeMaterialType($hBeamNode,
$IRR_EMT_TRANSPARENT_ALPHA_CHANNEL)
```

```
$hCamera = _IrrAddCamera(50, 0, 0, 0, 0, 0)
```

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
```

WEnd

```
_IrrStop()
Copy to Clipboard
```

IrrAddBoltSceneNode

The bolt is a special scene node that can be used to replicate electrical effects.

#Include <au3Irrlicht2.au3>
_IrrAddBoltSceneNode()

Parameters

None.

Return Value

Success: Handle to a Bolt node Failure: False and @error 1

Remarks

This command simply adds the bolt you should then make a call to set the bolts properties.

This node can be used to simulate lightning and other electrical effects.

Related

_IrrSetBoltProperties

Example

#include "au3Irrlicht2.au3"

Global \$hBoltNode, \$hTexture, \$hCamera

_IrrStart()

```
$hBoltNode = _IrrAddBoltSceneNode()
_IrrSetBoltProperties($hBoltNode, 0, 90, 0, 0, 0, 0, 50, 10, 2, 10, 6,
$IRR_ON, 0xFFFFFFF)
$hTexture = _IrrGetTexture("./media/ParticleBlue.bmp")
_IrrSetNodeMaterialTexture($hBoltNode, $hTexture, 0)
```

```
$hCamera = _IrrAddCamera(40, 50, 40, 0, 50, 0)
```

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(16, 24, 32)
_IrrDrawScene()
_IrrEndScene()
WEnd
```

_IrrStop()

```
Copy to Clipboard
```

IrrSetBoltProperties

This sets the properties of a bolt node that simulates an electrical effect.

#Include <au3Irrlicht2.au3>
_IrrSetBoltProperties(\$h_BoltNode, \$f_SX, \$f_SY, \$f_SZ, \$f_EX, \$f_EY,
\$f_EZ, \$i_UpdateTime=50, \$i_Height=10, \$f_Thickness=5.0, \$i_Parts=10,
\$i_Bolts=6, \$i_Steadyend=\$IRR_OFF, \$i_Color=0x0000FFFF)

Parameters

\$h_BoltNode	[explanation]
\$f_SX, \$f_SY, \$f_SZ	Starting point that the bolt origionates from.
\$f_EX, \$f_EY, \$f_EZ	End terminating point for the bolt.
\$i_UpdateTime	Number of miliseconds between updates to the appearence of the bolt.
\$i_Height	Radius is the radius of the entire bolt effect.
\$f_Thickness	Thickness of a single electrical element in the bolt.
\$i_Parts	Number of segments the bolt is divided into.
\$i_Bolts	Number of individual electrical arcs that are rendered.
\$i_Steadyend	Set to IRR_ON ends in a tight point, Set to IRR_OFF it ends with the same width as the rest of the bolt.
\$i_Color	Diffuse color that is applied to the bolt.

Return Value

Success: True Failure: False

Remarks

There are a number of properties that control many aspects of the bolt to produce a wide range of appearences.

Related

```
_IrrAddBoltSceneNode
```

Example

```
#include "au3Irrlicht2.au3"
```

Global \$hBoltNode, \$hTexture, \$hCamera

_IrrStart()

```
$hBoltNode = _IrrAddBoltSceneNode()
_IrrSetBoltProperties($hBoltNode, 0, 90, 0, 0, 0, 0, 50, 10, 2, 10, 6,
$IRR_ON, 0xFFFFFFF)
$hTexture = _IrrGetTexture("./media/ParticleBlue.bmp")
_IrrSetNodeMaterialTexture($hBoltNode, $hTexture, 0)
```

```
$hCamera = _IrrAddCamera(40, 50, 40, 0, 50, 0)
```

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(16, 24, 32)
_IrrDrawScene()
_IrrEndScene()
```

WEnd

```
_IrrStop()
Copy to Clipboard
```

IrrSetBillBoardColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetBillBoardColor(\$h_Node, \$i_TopColor, \$i_BottomColor)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

[todo: functionName, functionName]

IrrAddBillBoardGroupToScene

Adds a billboard group to the scene.

#Include <au3Irrlicht2.au3>
_IrrAddBillBoardGroupToScene()

Parameters

None.

Return Value

Success: Handle to an irr node object. Failure: False and @error 1

Remarks

This is a special object that can have billboard like objects added and removed from it and rendered in a very quick an efficient manner. They are all treated as a single object rather than as many individual nodes. This is particuallarly useful for custom particle effects.

Related

<u>_IrrAddBillBoardToGroup</u>, <u>_IrrRemoveBillBoardFromGroup</u>, <u>_IrrAddBillBoardByAxisToGroup</u>, <u>_IrrGetBillBoardGroupCount</u>, <u>IrrBillBoardGroupShadows</u>, <u>_IrrBillBoardForceUpdate</u>

Example

```
#include "au3Irrlicht2.au3"
```

Global \$hBillboardGroup Global \$hBillboardTexture Global \$aBillboard[6] Global \$hCamera

_IrrStart()

```
$hBillboardGroup = _IrrAddBillBoardGroupToScene()
For $i = 1 To 5
    $aBillboard[$i] = _IrrAddBillBoardToGroup($hBillboardGroup, 150, 100,
-450 + ($i * 150), 0, 400, 0, 255, 255, 255, 255)
Next
$hBillboardTexture = _IrrGetTexture("./media/au3irr2_logo.jpg")
_IrrSetNodeMaterialTexture($hBillboardGroup, $hBillboardTexture, 0)
_IrrSetNodeMaterialFlag($hBillboardGroup, $IRR_EMF_LIGHTING,
$IRR_OFF)
```

```
$hCamera = _IrrAddFPSCamera($IRR_NO_OBJECT, 100.0, 0.1)
_IrrSetCameraClipDistance($hCamera, 128000)
```

_IrrHideMouse()

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(0, 0, 255)
_IrrDrawScene()
_IrrEndScene()
WEnd
Copy to Clipboard
```

IrrAddBillBoardToGroup

Adds a billboard to a billboard group. There are a number of properties that are used to specify the billboard

#Include <au3Irrlicht2.au3>
_IrrAddBillBoardToGroup(\$h_BillboardGroup, \$f_XSize, \$f_YSize,
\$f_XPos, \$f_YPos, \$f_ZPos, \$f_Roll, \$u_Alpha, \$u_Red , \$u_Green,
\$u_Blue)

Parameters

\$h_BillboardGroup	Handle to the Billboard Group as returned by _IrrAddBillBoardGroupToScene
\$f_XSize	Width of the billboard
\$f_YSize	Height of the billboard
\$f_XPos	X position of the billboard
\$f_YPos	Y position of the billboard
\$f_ZPos	Z position of the billboard
\$f_Roll	Specifies the number of degrees that the billboard is spun around its center
\$u_Alpha	Alpha color used for the billboard 0 - 255
\$u_Red	Red color used for the billboard 0 - 255
\$u_Green	Green color used for the billboard 0 - 255
\$u_Blue	Blue color used for the billboard 0 - 255

Return Value

Success: Handle to the billbord scene node address in the billbord group Failure: False and @error 1 $\,$

Remarks

None

Related

```
_IrrAddBillBoardGroupToScene, _IrrRemoveBillBoardFromGroup,
_IrrGetBillBoardGroupCount, _IrrAddBillBoardByAxisToGroup,
_IrrBillBoardForceUpdate
```

Example

```
#include "au3Irrlicht2.au3"
Global $hBillboardGroup
Global $hBillboardTexture
Global $aBillboard[6]
Global $hCamera
_IrrStart()
$hBillboardGroup = IrrAddBillBoardGroupToScene()
For $i = 1 To 5
  $aBillboard[$i] = IrrAddBillBoardToGroup($hBillboardGroup, 150, 100,
-450 + ($i * 150), 0, 400, 0, 255, 255, 255, 255)
Next
$hBillboardTexture = _IrrGetTexture("./media/au3irr2_logo.jpg")
_IrrSetNodeMaterialTexture($hBillboardGroup, $hBillboardTexture, 0)
_IrrSetNodeMaterialFlag($hBillboardGroup, $IRR_EMF_LIGHTING,
$IRR OFF)
$hCamera = _IrrAddFPSCamera($IRR_NO_OBJECT, 100.0, 0.1)
IrrSetCameraClipDistance($hCamera, 128000)
```

```
_IrrHideMouse()
```

While _IrrRunning() And Sleep(10) _IrrBeginScene(0, 0, 255) _IrrDrawScene() _IrrEndScene() WEnd Copy to Clipboard

IrrAddBillBoardByAxisToGroup

Adds a billboard to a billboard group that is fixed to a particular axis.

#Include <au3Irrlicht2.au3>
_IrrAddBillBoardByAxisToGroup(\$h_BillboardGroup, \$f_XSize, \$f_YSize,
\$f_XPos, \$f_YPos, \$f_ZPos, \$f_Roll, \$u_Alpha, \$u_Red, \$u_Green,
\$u_Blue, \$f_XAxis, \$f_YAxis, \$f_ZAxis)

Parameters

\$h_BillboardGroup	Handle to the Billboard Group as returned by _IrrAddBillBoardGroupToScene
\$f_XSize	Width of the billboard
\$f_YSize	Height of the billboard
\$f_XPos	X position of the billboard
\$f_YPos	Y position of the billboard
\$f_ZPos	Z position of the billboard
\$f_Roll	Specifies the number of degrees that the billboard is spun around its center
\$u_Alpha	Alpha color used for the billboard 0 - 255
\$u_Red	Red color used for the billboard 0 - 255
\$u_Green	Green color used for the billboard 0 - 255
\$u_Blue	Blue color used for the billboard 0 - 255
\$f_XAxis	X direction around which the billboard is spun to face the camera.
\$f_YAxis	Y direction around which the billboard is spun to face the camera.
\$f_ZAxis	Z direction around which the billboard is spun to face the camera

Return Value

Success: Handle to the billbord scene node address in the billbord group Failure: False and @error 1 $\,$

Remarks

These billboards are particularly useful for things like grass. There are a number of properties that are used to specify the billboard.

Related

```
<u>_IrrAddBillBoardGroupToScene</u>, <u>_IrrRemoveBillBoardFromGroup</u>,
<u>_IrrGetBillBoardGroupCount</u>, <u>_IrrBillBoardForceUpdate</u>
```

Example

#include "au3Irrlicht2.au3"

Global \$hBillboardGroup Global \$hBillboardTexture Global \$aBillboard[6] Global \$hCamera

_IrrStart()

```
$hBillboardGroup = _IrrAddBillBoardGroupToScene()
For $i = 1 To 5
    $aBillboard[$i] = _IrrAddBillBoardByAxisToGroup($hBillboardGroup,
150, 100, -450 + ($i * 150), 0, 400, 0, 255, 255, 255, 255, 0, 1, 0)
Next
$hBillboardTexture = _IrrGetTexture("./media/au3irr2_logo.jpg")
_IrrSetNodeMaterialTexture($hBillboardGroup, $hBillboardTexture, 0)
IrrSetNodeMaterialFlag($hBillboardGroup, $IRR_EMF_LIGHTING,
```

\$IRR_OFF)

```
$hCamera = _IrrAddFPSCamera($IRR_NO_OBJECT, 100.0, 0.1)
_IrrSetCameraClipDistance($hCamera, 128000)
```

_IrrHideMouse()

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(0, 0, 255)
_IrrDrawScene()
_IrrEndScene()
WEnd
```

Copy to Clipboard

IrrRemoveBillBoardFromGroup

Removes the specified billboard from the billboard group.

#Include <au3Irrlicht2.au3>
_IrrRemoveBillBoardFromGroup(\$h_BillboardGroup,
\$h_BillboardGroupSceneNode)

Parameters

\$h_BillboardGroup	Handle to the Billboard Group as returned by _IrrAddBillBoardGroupToScene
\$h_BillboardGroupSceneNode	Handle to the billbord scene node address in the billbord group as returned by _IrrAddBillBoardToGroup

Return Value

Success: True Failure: False

Remarks

None.

Related

_IrrAddBillBoardGroupToScene, _IrrAddBillBoardToGroup

Example

#include "au3Irrlicht2.au3"

```
Global $hBillboardGroup
Global $hBillboardTexture
Global $aBillboard[6]
Global $iBillBoards
Global $hCamera
Global $iTimer = TimerInit()
```

_IrrStart()

```
$hCamera = _IrrAddFPSCamera($IRR_NO_OBJECT, 100.0, 0.1)
_IrrSetCameraClipDistance($hCamera, 128000)
```

```
_IrrHideMouse()
```

```
While _IrrRunning() And Sleep(10)
  $iBillBoards = _IrrGetBillBoardGroupCount($hBillboardGroup)
  If TimerDiff($iTimer) > 2000 And $iBillBoards > 0 Then
  _IrrRemoveBillBoardFromGroup($hBillboardGroup,
$aBillboard[$iBillBoards])
  $iTimer = TimerInit()
  EndIf
  _IrrBeginScene(0, 0, 255)
  _IrrDrawScene()
   IrrEndScene()
```

WEnd

Copy to Clipboard

IrrBillBoardGroupShadows

Applies lighting to the billboards in a cluster of billboards.

#Include <au3Irrlicht2.au3>
_IrrBillBoardGroupShadows(\$h_BillboardGroup, \$f_X, \$f_Y, \$f_Z,
\$f_Intensity, \$f_Ambient)

Parameters

\$h_BillboardGroup	Handle to the Billboard Group as returned by _IrrAddBillBoardGroupToScene
\$f_X	X direction from which the light is arriving.
\$f_Y	Y direction from which the light is arriving.
\$f_Z	Z direction from which the light is arriving.
\$f_Intensity	Strength of the light
\$f_Ambient	Strength of ambient light in the billboard group

Return Value

Success: True Failure: False

Remarks

This can be used for example to shade the particles in a group of billboards representing a cloud.

Related

_IrrAddBillBoardGroupToScene, _IrrAddBillBoardToGroup

Example

```
_IrrSetNodeMaterialFlag($hBillboardGroup, $IRR_EMF_LIGHTING,
$IRR_OFF)
```

```
$hCamera = _IrrAddFPSCamera($IRR_NO_OBJECT, 100.0, 0.1)
_IrrSetCameraClipDistance($hCamera, 128000)
```

```
_IrrHideMouse()
```

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(0, 0, 255)
_IrrDrawScene()
_IrrEndScene()
WEnd
```

Copy to Clipboard

IrrGetBillBoardGroupCount

Get the number of billboards in the billboard group.

#Include <au3Irrlicht2.au3>
_IrrGetBillBoardGroupCount(\$h_BillboardGroup)

Parameters

th DillboordCroup	Handle to the Billboard Group as returned by
\$11_DiliboardGroup	Handle to the Billboard Group as returned by _IrrAddBillBoardGroupToScene

Return Value

Success: The number of billboards in the billboard group Failure: False and @error 1

Remarks

None.

Related

<u>_IrrAddBillBoardGroupToScene</u>, <u>_IrrAddBillBoardToGroup</u>, <u>_IrrRemoveBillBoardFromGroup</u>

Example

#include "au3Irrlicht2.au3"

```
Global $hBillboardGroup
Global $hBillboardTexture
Global $aBillboard[6]
Global $hCamera
```

_IrrStart()

```
_IrrSetWindowCaption(_IrrGetBillBoardGroupCount($hBillboardGroup) & "
Billboards in the Billboard Group")
```

```
$hCamera = _IrrAddFPSCamera($IRR_NO_OBJECT, 100.0, 0.1)
_IrrSetCameraClipDistance($hCamera, 128000)
```

```
_IrrHideMouse()
```

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(0, 0, 255)
_IrrDrawScene()
_IrrEndScene()
WEnd
```

Copy to Clipboard

IrrBillBoardForceUpdate

Force the billboard group update the scene. (see remarks for further explanation)

#Include <au3Irrlicht2.au3>
_IrrBillBoardForceUpdate(\$h_BillboardGroup)

Parameters

\$h_BillboardGroup	Handle to the Billboard Group as returned by _IrrAddBillBoardGroupToScene
	_IrrAddBillBoardGroupToScene

Return Value

Success: True Failure: False

Remarks

Unlike regular billboards the billboard group does not always update the orientation of the billboards every frame.

If you are a long distance away from the billboard group the camera needs to travel a significant distance before the

angle has changed enough to warrent an update of all of the billboards verticies to make them point to the camera once more.

You may want to force a refresh at some point with this call.

Related

<u>_IrrAddBillBoardGroupToScene</u>, <u>_IrrAddBillBoardToGroup</u>, <u>_IrrAddBillBoardByAxisToGroup</u>, <u>_IrrRemoveBillBoardFromGroup</u>

IrrAddLODManager

Adds a level of detail manager to the scene.

#Include <au3Irrlicht2.au3>
_IrrAddLODManager(\$u_FadeScale, \$u_UseAlpha[, \$p_Callback = 0])

Parameters

\$u_FadeScale	Number of 1/4 seconds that the node takes to fade out or in. 4 units equals 1 second.
\$u_UseAlpha	Specifies whether or not the Alpha color of the object is faded too.
\$p_Callback	[optional] Register a callback function that is called whenever a node is made invisible or visible. This allows you to stop processing hidden nodes.

Return Value

Success: Handle to the LOD Manager node Failure: False and @error 1

Remarks

The primary use for this node is to add other scene nodes to it as children and have their level of detail controlled automatically. If those nodes are made from loaded meshs different meshes containing different amounts of detail can be displayed at different distances. The other function of the LOD manager is to fade nodes in an out at a specific distance so they gradually fade rather than disappear abruptly. This is achieved by applying a distance without supplying a mesh.

Related

_IrrAddLODMesh, _IrrSetLODMaterialMap

Example

#include "au3Irrlicht2.au3"

Global \$ROWS_AND_COLUMNS = 20
Global \$hLOD1Mesh
Global \$hLOD2Mesh
Global \$hMeshTexture
<pre>Global \$iAmountNodes = \$ROWS_AND_COLUMNS *</pre>
\$ROWS_AND_COLUMNS
Global \$aSceneNodes[\$iAmountNodes]
Global \$hMaterial
Global $k = 0$
Global \$hLODManager
Global \$hCamera

_IrrStart()

```
$hLOD1Mesh = _IrrGetMesh("./media/cylinderY.obj")
$hLOD2Mesh = _IrrGetMesh("./media/cylinderYLow.obj")
_IrrScaleMesh($hLOD1Mesh, 8.0)
_IrrScaleMesh($hLOD2Mesh, 8.0)
_IrrSetMeshHardwareAccelerated($hLOD1Mesh)
_IrrSetMeshHardwareAccelerated($hLOD2Mesh)
$hMeshTexture = _IrrGetTexture("./media/Cross.bmp")
For $i = -($ROWS_AND_COLUMNS / 2) To ($ROWS_AND_COLUMNS /
2) - 1
For $j = -($ROWS_AND_COLUMNS / 2) To ($ROWS_AND_COLUMNS /
2) - 1
For $j = -($ROWS_AND_COLUMNS / 2) To ($ROWS_AND_COLUMNS /
2) - 1
saSceneNodes[$k] = _IrrAddMeshToScene($hLOD1Mesh)
_IrrSetNodePosition($aSceneNodes[$k], $i * 40.0, 0.0, $j * 40.0)
```

_IrrSetNodeMaterialTexture(\$aSceneNodes[\$k], \$hMeshTexture, 0) _IrrSetNodeMaterialFlag(\$aSceneNodes[\$k], \$IRR_EMF_LIGHTING, \$IRR_ON)

\$hMaterial = _IrrGetMaterial(\$aSceneNodes[\$k], 0)
_IrrMaterialVertexColorAffects(\$hMaterial, \$ECM_NONE)
_IrrMaterialSetAmbientColor(\$hMaterial, 255, 255, 255, 255)
_IrrMaterialSetDiffuseColor(\$hMaterial, 255, 255, 255, 255)
\$k += 1

Next

Next

```
$hLODManager = _IrrAddLODManager(2, $IRR_ON)
_IrrSetLODMaterialMap($hLODManager,
$IRR_EMT_TRANSPARENT_ADD_COLOR,
$IRR_EMT_TRANSPARENT_ADD_COLOR)
_IrrAddLODMesh($hLODManager, 0.0, $hLOD1Mesh)
_IrrAddLODMesh($hLODManager, 300.0, $hLOD2Mesh)
_IrrAddLODMesh($hLODManager, 500.0, $IRR_NO_OBJECT)
_IrrSetNodeMaterialFlag($hLODManager, $IRR_EMF_LIGHTING,
$IRR_OFF)
For $i = 0 To $k - 1
```

_IrrAddChildToParent(\$aSceneNodes[\$i], \$hLODManager) Next

```
$hCamera = _IrrAddFPSCamera($IRR_NO_OBJECT, 100.0, 0.05)
_IrrSetNodePosition($hCamera, 0, $ROWS_AND_COLUMNS * 4,
$ROWS_AND_COLUMNS * 2)
_IrrSetCameraTarget($hCamera, 0, 50, 0)
```

```
_IrrSetCameraClipDistance($hCamera, 2500.0)
```

_IrrSetAmbientLight(1, 1, 1)

_IrrHideMouse()

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
```

WEnd

_IrrStop()
Copy to Clipboard

IrrAddLODMesh

Set the distance at which a particular mesh is to be applied to child mesh nodes.

#Include <au3Irrlicht2.au3>
_IrrAddLODMesh(\$h_LODManager, \$f_Distance, \$h_Mesh)

Parameters

\$h_LODManager	Handle to the LOD Manager node.
\$f_Distance	Distance at which this effect will be applied.
\$h_Mesh	Handle to an irr mesh object

Return Value

Success: True Failure: False

Remarks

If no mesh is supplied it specifies the distance at which the node should be faded in an out.

Related

_IrrAddLODManager, _IrrSetLODMaterialMap

Example

#include "au3Irrlicht2.au3"

```
Global $ROWS_AND_COLUMNS = 20
Global $hLOD1Mesh
Global $hLOD2Mesh
Global $hMeshTexture
Global $iAmountNodes = $ROWS_AND_COLUMNS *
$ROWS_AND_COLUMNS
Global $aSceneNodes[$iAmountNodes]
Global $hMaterial
Global $hMaterial
Global $hLODManager
Global $hLODManager
```

_IrrStart()

```
$hLOD1Mesh = _IrrGetMesh("./media/cylinderY.obj")
$hLOD2Mesh = _IrrGetMesh("./media/cylinderYLow.obj")
IrrScaleMesh($hLOD1Mesh, 8.0)
_IrrScaleMesh($hLOD2Mesh, 8.0)
_IrrSetMeshHardwareAccelerated($hLOD1Mesh)
_IrrSetMeshHardwareAccelerated($hLOD2Mesh)
$hMeshTexture = _IrrGetTexture("./media/Cross.bmp")
For $i = -($ROWS AND COLUMNS / 2) To ($ROWS AND COLUMNS /
2) - 1
  For $j = -($ROWS AND COLUMNS / 2) To ($ROWS AND COLUMNS
/2)-1
    $aSceneNodes[$k] = _IrrAddMeshToScene($hLOD1Mesh)
    _IrrSetNodePosition($aSceneNodes[$k], $i * 40.0, 0.0, $j * 40.0)
    IrrSetNodeMaterialTexture($aSceneNodes[$k], $hMeshTexture, 0)
    IrrSetNodeMaterialFlag($aSceneNodes[$k], $IRR EMF LIGHTING,
$IRR ON)
    $hMaterial = IrrGetMaterial($aSceneNodes[$k], 0)
    IrrMaterialVertexColorAffects($hMaterial, $ECM NONE)
    IrrMaterialSetAmbientColor($hMaterial, 255, 255, 255, 255)
    _IrrMaterialSetDiffuseColor($hMaterial, 255, 255, 255, 255)
    $k += 1
```

Next

Next

```
$hLODManager = _IrrAddLODManager(2, $IRR_ON)
_IrrSetLODMaterialMap($hLODManager,
$IRR EMT TRANSPARENT ADD COLOR,
$IRR_EMT_TRANSPARENT_ADD_COLOR)
_IrrAddLODMesh($hLODManager, 0.0, $hLOD1Mesh)
_IrrAddLODMesh($hLODManager, 300.0, $hLOD2Mesh)
IrrAddLODMesh($hLODManager, 500.0, $IRR NO OBJECT)
IrrSetNodeMaterialFlag($hLODManager, $IRR EMF LIGHTING,
$IRR OFF)
For $i = 0 To $k - 1
  _IrrAddChildToParent($aSceneNodes[$i], $hLODManager)
Next
$hCamera = _IrrAddFPSCamera($IRR_NO_OBJECT, 100.0, 0.05)
IrrSetNodePosition($hCamera, 0, $ROWS AND COLUMNS * 4,
$ROWS AND COLUMNS * 2)
_IrrSetCameraTarget($hCamera, 0, 50, 0)
_IrrSetCameraClipDistance($hCamera, 2500.0)
_IrrSetAmbientLight(1, 1, 1)
_IrrHideMouse()
While IrrRunning() And Sleep(10)
  _IrrBeginScene(0, 0, 0)
  _IrrDrawScene()
  _IrrEndScene()
WEnd
IrrStop()
```

Copy to Clipboard

IrrSetLODMaterialMap

Specifies which material is used to apply the fade effect for another material type.

#Include <au3Irrlicht2.au3>
_IrrSetLODMaterialMap(\$h_LODManager, \$i_SourceType, \$i_TargetType)

Parameters

\$h_LODManager	Handle to the LOD Manager node.
\$u_SourceType	The irr material type your node uses
\$u_TargetType	The material type used for the fade effect.

Return Value

Success: True Failure: False

Remarks

How this is used will depend on the effect that you want to achieve. By default fading is applied with the \$IRR_EMT_TRANSPARENT_VERTEX_ALPHA material.

Related

_IrrAddLODManager, _IrrAddLODMesh

Example

#include "au3Irrlicht2.au3"

Global \$ROWS_AND_COLUMNS = 20 Global \$hLOD1Mesh Global \$hLOD2Mesh Global \$hMeshTexture Global \$iAmountNodes = \$ROWS_AND_COLUMNS * \$ROWS_AND_COLUMNS Global \$aSceneNodes[\$iAmountNodes] Global \$hMaterial Global \$hMaterial Global \$k = 0 Global \$hLODManager Global \$hCamera

_IrrStart()

```
$hLOD1Mesh = _IrrGetMesh("./media/cylinderY.obj")
```

```
$hLOD2Mesh = _IrrGetMesh("./media/cylinderYLow.obj")
```

```
_IrrScaleMesh($hLOD1Mesh, 8.0)
```

```
_IrrScaleMesh($hLOD2Mesh, 8.0)
```

_IrrSetMeshHardwareAccelerated(\$hLOD1Mesh)

```
_IrrSetMeshHardwareAccelerated($hLOD2Mesh)
```

```
$hMeshTexture = _IrrGetTexture("./media/Cross.bmp")
```

For \$i = -(\$ROWS_AND_COLUMNS / 2) To (\$ROWS_AND_COLUMNS /
2) - 1

```
For $j = -($ROWS_AND_COLUMNS / 2) To ($ROWS_AND_COLUMNS / 2) - 1
```

```
$aSceneNodes[$k] = _IrrAddMeshToScene($hLOD1Mesh)
```

_IrrSetNodePosition(\$aSceneNodes[\$k], \$i * 40.0, 0.0, \$j * 40.0)

```
_IrrSetNodeMaterialTexture($aSceneNodes[$k], $hMeshTexture, 0)
```

```
_IrrSetNodeMaterialFlag($aSceneNodes[$k], $IRR_EMF_LIGHTING, $IRR_ON)
```

```
$hMaterial = _IrrGetMaterial($aSceneNodes[$k], 0)
```

```
_IrrMaterialVertexColorAffects($hMaterial, $ECM_NONE)
```

```
_IrrMaterialSetAmbientColor($hMaterial, 255, 255, 255, 255)
```

```
_IrrMaterialSetDiffuseColor($hMaterial, 255, 255, 255, 255)
```

```
$k += 1
```

Next

Next

```
$hLODManager = _IrrAddLODManager(2, $IRR_ON)
_IrrSetLODMaterialMap($hLODManager,
$IRR_EMT_TRANSPARENT_ADD_COLOR,
$IRR_EMT_TRANSPARENT_ADD_COLOR)
_IrrAddLODMesh($hLODManager, 0.0, $hLOD1Mesh)
_IrrAddLODMesh($hLODManager, 300.0, $hLOD2Mesh)
_IrrAddLODMesh($hLODManager, 500.0, $IRR_NO_OBJECT)
_IrrSetNodeMaterialFlag($hLODManager, $IRR_EMF_LIGHTING,
$IRR_OFF)
For $i = 0 To $k - 1
_IrrAddChildToParent($aSceneNodes[$i], $hLODManager)
```

Next

```
$hCamera = _IrrAddFPSCamera($IRR_NO_OBJECT, 100.0, 0.05)
_IrrSetNodePosition($hCamera, 0, $ROWS_AND_COLUMNS * 4,
$ROWS_AND_COLUMNS * 2)
_IrrSetCameraTarget($hCamera, 0, 50, 0)
```

```
_IrrSetCameraClipDistance($hCamera, 2500.0)
```

_IrrSetAmbientLight(1, 1, 1)

_IrrHideMouse()

```
While _IrrRunning() And Sleep(10)
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEnd
```

```
_IrrStop()
Copy to Clipboard
```

_IrrStart

Opens the IrrlichtWrapper.dll, starts Irrlicht interface and opens a window for rendering.

```
#Include <au3Irrlicht2.au3>
_IrrStart($i_DeviceType=$IRR_EDT_DIRECT3D9, $i_ScreenWidth=800,
$i_ScreenHeight=600, $i_BitsPerPixel=$IRR_BITS_PER_PIXEL_32,
$b_FullScreen=$IRR_WINDOWED, $b_Shadows=$IRR_NO_SHADOWS,
$b_InputCapture=$IRR_IGNORE_EVENTS,
$b_VSync=$IRR_VERTICAL_SYNC_OFF)
```

Parameters

\$i_DeviceType	[optional] specifies the renderer to use when drawing to the display this may be one of the following types: \$IRR_EDT_NULL - A NULL device with no display \$IRR_EDT_SOFTWARE - Irrlichts default software renderer \$IRR_EDT_SOFTWARE2 - An improved quality software renderer \$IRR_EDT_OPENGL - Hardware accelerated OpenGL renderer \$IRR_EDT_DIRECT3D8 - Hardware accelerated DirectX 8 renderer \$IRR_EDT_DIRECT3D9 - Hardware accelerated DirectX 9 renderer
\$i_ScreenWidth	[optional] Screen width specifies the width of the viewport in pixels
\$i_ScreenHeight	[optional] Screen height specifies the height of the viewport in pixels
	[optional] The number of color bits that is used for each pixel 32 bit color gives 24 million different colors

\$i_BitsPerPixel	whereas 16 bit color gives only 32,000 colors. However the advantage of 16 bit color is that some operations use half the memory and can run at up to twice the speed. This setting can be either of: \$IRR_BITS_PER_PIXEL_16 \$IRR_BITS_PER_PIXEL_32
\$b_FullScreen	[optional] Specifies whether the display is to opened in full screen mode or in a window: \$IRR_WINDOWED - For window mode \$IRR_FULLSCREEN - For fullscreen mode. When using full screen mode you will need to adjust the window size to the same dimensions as a supported screen resolution on the target display 640x400 for example.
\$b_Shadows	[optional] Use shadows starts the engine in a mode that supports the rendering of stencil shadows. \$IRR_NO_SHADOWS - For a display that does not support shadows. \$IRR_SHADOWS - For a display that supports shadows.
\$b_InputCapture	[optional] Capture mouse and keyboard specified whether you want to capture keyboard and mouse events, if you choose to ignore them they will be handled by Irrlicht for FPS camera control. This parameter should be either of: \$IRR_IGNORE_EVENTS \$IRR_CAPTURE_EVENTS
\$b_VSync	[optional] Vertical syncronisation specifies whether the display of each new frame is syncronised with vertical refresh of the graphics card. This produces a smoother display and avoids 'tearing' where the viewer can see parts of two different frames at the same time. The setting can be either of: \$IRR_VERTICAL_SYNC_OFF \$IRR_VERTICAL_SYNC_ON

Return Value

Success: True Failure: False and sets @error:

- 1 error occured on dll call
- 2 IrrlichtWrapper.dll not found

Remarks

if .dll cannot be opened, path environment is extended with:

- .\bin (allows an au3Irr2 script to have its binaries in a subdir) and

- .1.. (allows au3Irr2 examples to be started from their \include subdir). Nevertheless, ensuring DLL's in a permanent dir reachable via path may be the better way, as the temporary update of environment can be time-consuming.

Other needed .dll's (Irrlicht.dll + maybe msvcp71.dll, msvcr71.dll) **are NOT checked** but simply expected to be at last in same dir as the IrrlichtWrapper.dll.

Related

_IrrStartAdvanced, _IrrRunning, _IrrStop

Example

#include "au3Irrlicht2.au3"

_IrrStart(\$IRR_EDT_OPENGL, 800, 600, \$IRR_BITS_PER_PIXEL_32, _ \$IRR_WINDOWED, \$IRR_SHADOWS, \$IRR_CAPTURE_EVENTS, \$IRR_VERTICAL_SYNC_ON)

local \$Camera = _IrrAddCamera(10,10,10, 0,0,0)
local \$testNode = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture(\$testNode,

```
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $testNode, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrStartAdvanced

Opens the IrrlichtWrapper.dll and starts Irrlicht engine with advanced method.

```
#Include <au3Irrlicht2.au3>
_IrrStartAdvanced($i_DeviceType=$IRR_EDT_DIRECT3D9,
$i_ScreenWidth=800, $i_ScreenHeight=600,
$i_BitsPerPixel=$IRR_BITS_PER_PIXEL_32,
$b_FullScreen=$IRR_WINDOWED, $b_Shadows=$IRR_NO_SHADOWS,
$b_InputCapture=$IRR_IGNORE_EVENTS,
$b_VSync=$IRR_VERTICAL_SYNC_OFF, $i_TypeOfDevice=0,
$b_DoublebufferEnabled=$IRR_OFF, $i_AntialiasEnabled=0,
$b_HighPrecisionFpu=$IRR_OFF)
```

Parameters

\$i_DeviceType	 [optional] specifies the renderer to use when drawing to the display this may be one of the following types: \$IRR_EDT_NULL - A NULL device with no display \$IRR_EDT_SOFTWARE - Irrlichts default software renderer \$IRR_EDT_SOFTWARE2 - An improved quality software renderer \$IRR_EDT_OPENGL - Hardware accelerated OpenGL renderer \$IRR_EDT_DIRECT3D8 - Hardware accelerated DirectX 8 renderer
	accelerated DirectX 8 renderer \$IRR_EDT_DIRECT3D9 - Hardware
\$i_ScreenWidth	accelerated DirectX 9 renderer [optional] Screen width specifies the width of the viewport in pixels

\$i_ScreenHeight	[optional] Screen height specifies the height of the viewport in pixels
\$i_BitsPerPixel	[optional] The number of color bits that is used for each pixel 32 bit color gives 24 million different colors whereas 16 bit color gives only 32,000 colors. However the advantage of 16 bit color is that some operations use half the memory and can ru at up to twice the speed. This setting can be either of: \$IRR_BITS_PER_PIXEL_16 \$IRR_BITS_PER_PIXEL_32
\$b_FullScreen	[optional] Specifies whether the display is opened in full screen mode or in a window: \$IRR_WINDOWED - For window mode \$IRR_FULLSCREEN - For fullscreen mode When using full screen mode you will need adjust the window size to the same dimensions as a supported screen resolution on the target display 640x400 for example.
\$b_Shadows	[optional] Use shadows starts the engine in mode that supports the rendering of stencil shadows. \$IRR_NO_SHADOWS - For a display that does not support shadows. \$IRR_SHADOWS - For a display that supports shadows.
\$b_InputCapture	[optional] Capture mouse and keyboard specified whether you want to capture keyboard and mouse events, if you choose ignore them they will be handled by Irrlicht FPS camera control. This parameter should be either of: \$IRR_IGNORE_EVENTS \$IRR_CAPTURE_EVENTS
	[optional] Vertical syncronisation specifies whether the display of each new frame is

\$b_VSync	syncronised with vertical refresh of the graphics card. This produces a smoother display and avoids 'tearing' where the viewer can see parts of two different frames at the same time. The setting can be either of: \$IRR_VERTICAL_SYNC_OFF \$IRR_VERTICAL_SYNC_ON
\$i_TypeOfDevice	[optional] Devicetype allows a specific type of device for example a windows screen or a console to be selected. For the time being this should be set to 0 which automatically selects the best device.
\$b_DoublebufferEnabled	[optional] Doublebufferenabled is used to control whether double buffering is used. When double buffering is used two drawing surfaces are created one for display and the other that is used for drawing too. Double buffering is required for anit-aliasing the options are: \$IRR_ON or \$IRR_OFF
\$i_AntialiasEnabled	[optional] Antialiasenabled is used to enable the antialiasing effect, this effect produces a blurring at the edges of object giving their lines a smooth natural appearence. There is usually a big penalty for using this effect though sometimes as high as 30% of the frame rate or more. This is a value for the anti-aliasing and should be a power of 2. (e.g: 2, 4, 8, 16)
\$b_HighPrecisionFpu	[optional] Highprecisionfpu is used to enable high precision Floating point calculations, that produce more accurate result at the expense of a slower operating speed.

Return Value

Success: True

Failure: False and sets @error:

- 1 error occured on dll call
- 2 IrrlichtWrapper.dll not found

Remarks

if .dll cannot be opened, path environment is extended with:

- **.\bin** (allows an au3Irr2 script to have its binaries in a subdir) and - **.\.** (allows au3Irr2 examples to be started from their \include subdir). Nevertheless, ensuring DLL's in a permanent dir reachable via path may be the better way, as the temporary update of environment can be timeconsuming.

Other needed .dll's (Irrlicht.dll + maybe msvcp71.dll, msvcr71.dll) **are NOT checked** but simply expected to be at last in same dir as the IrrlichtWrapper.dll.

Related

_IrrStart, _IrrRunning, _IrrStop

Example

#include "au3Irrlicht2.au3"

```
_IrrStartAdvanced ( $IRR_EDT_OPENGL, 800, 600,
$IRR_BITS_PER_PIXEL_32, _
$IRR_WINDOWED, $IRR_SHADOWS, $IRR_CAPTURE_EVENTS,
$IRR_VERTICAL_SYNC_ON, _
0, $IRR_ON, 4, $IRR_ON)
local $Camera = _IrrAddCamera(10,10,10, 0,0,0 )
local $testNode = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $testNode,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
```

_IrrSetNodeMaterialFlag(\$testNode, \$IRR_EMF_LIGHTING, \$IRR_OFF)

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrRunning

Used to determine if the Irrlicht engine is still running.

#Include <au3Irrlicht2.au3>
_IrrRunning()

Parameters

None.

Return Value

True if running: False if not. Sets @error to true on failures.

Remarks

None

Related

_IrrStart, _IrrStartAdvanced, _IrrStop

Example

#include "au3Irrlicht2.au3"

_IrrStart(\$IRR_EDT_OPENGL, 800, 600, \$IRR_BITS_PER_PIXEL_32, _____ \$IRR_WINDOWED, \$IRR_SHADOWS, \$IRR_CAPTURE_EVENTS,

```
$IRR_VERTICAL_SYNC_ON )
```

```
local $Camera = _IrrAddCamera(10,10,10, 0,0,0 )
local $testNode = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $testNode,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $testNode, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
```

WEND

_IrrStop()

```
Copy to Clipboard
```

IrrSetViewPort

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetViewPort(\$i_TopX, \$i_TopY, \$i_BottomX, \$i_BottomY)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrBeginScene

Starts to draw a frame, erasing the canvas with the specified color.

#Include <au3Irrlicht2.au3>
_IrrBeginScene(\$i_Red, \$i_Green, \$i_Blue)

Parameters

\$i_Red	Red value from 0 to 255 (full intensity)
\$i_Green	Green value from 0 to 255 (full intensity)
\$i_Blue	Blue value from 0 to 255 (full intensity)

Return Value

Success: True Failure: False

Remarks

None

Related

<u>IrrBeginSceneAdvanced</u>, <u>IrrDrawScene</u>, <u>IrrDrawGUI</u>, <u>IrrEndScene</u>

Example

#include "au3Irrlicht2.au3"

```
_IrrStart( $IRR_EDT_OPENGL, 800, 600, $IRR_BITS_PER_PIXEL_32, _
$IRR_WINDOWED, $IRR_SHADOWS, $IRR_CAPTURE_EVENTS,
$IRR_VERTICAL_SYNC_ON )
```

```
local $Camera = _IrrAddCamera(10,10,10, 0,0,0 )
local $testNode = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $testNode,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $testNode, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
```

```
_IrrStop()
```

```
Copy to Clipboard
```

IrrBeginSceneAdvanced

[todo]

#Include <au3Irrlicht2.au3>
_IrrBeginSceneAdvanced(\$i_SceneBGColor, \$b_ClearBackBuffer =
\$IRR_ON, \$b_ClearZBuffer = \$IRR_ON)

Parameters

Inaramii	[explanation] [moreTextForParam1]	
----------	--------------------------------------	--

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrDrawScene

This renders the 3D scene to the canvas, drawing all 3D elements: nodes, particles, billboards, etc .

#Include <au3Irrlicht2.au3>
_IrrDrawScene()

Parameters

None.

Return Value

Success: True Failure: False

Remarks

None

Related

_IrrBeginScene, _IrrBeginSceneAdvanced, _IrrDrawGUI, _IrrEndScene

Example

#include "au3Irrlicht2.au3"

_IrrStart(\$IRR_EDT_OPENGL, 800, 600, \$IRR_BITS_PER_PIXEL_32, _

```
$IRR_WINDOWED, $IRR_SHADOWS, $IRR_CAPTURE_EVENTS,
$IRR_VERTICAL_SYNC_ON )
```

```
local $Camera = _IrrAddCamera(10,10,10, 0,0,0 )
local $testNode = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $testNode,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $testNode, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
```

```
_IrrStop()
```

```
Copy to Clipboard
```

IrrDrawSceneToTexture

[todo]

#Include <au3Irrlicht2.au3>
_IrrDrawSceneToTexture(\$h_RenderTargetTexture)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetRenderTarget

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetRenderTarget(\$h_Texture, \$i_SceneBGColor = 0,
\$b_ClearBackBuffer = \$IRR_ON, \$b_ClearZBuffer = \$IRR_ON)

Parameters

[param1]	[explanation]
[param_]	[moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrDrawGUI

This renders the 2D graphical user interface that has been created to the scene.

#Include <au3Irrlicht2.au3> _IrrDrawGUI()

Parameters

None.	

Return Value

Success: True Failure: False

Remarks

None

Related

_IrrBeginScene, _IrrBeginSceneAdvanced, _IrrDrawScene, _IrrEndScene

Example

#include "au3Irrlicht2.au3"

```
_IrrStart( $IRR_EDT_OPENGL, 800, 600, $IRR_BITS_PER_PIXEL_32, _
$IRR_WINDOWED, $IRR_SHADOWS, $IRR_CAPTURE_EVENTS,
$IRR_VERTICAL_SYNC_ON )
```

```
_IrrAddStaticText( "Hello AU3-World ;-)", 8, 8, 200, 20,
$IRR_GUI_NO_BORDER, $IRR_GUI_NO_WRAP)
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawGUI()
_IrrEndScene()
WEND
```

```
_IrrStop()
```

Copy to Clipboard

IrrEndScene

This renders the 3D scene to the canvas, drawing all 3D elements: nodes, particles, billboards, etc .

#Include <au3Irrlicht2.au3>
_IrrEndScene()

Parameters

Return Value

Success: True Failure: False

Remarks

None

Related

_IrrBeginScene, _IrrBeginSceneAdvanced, _IrrDrawScene, _IrrDrawGUI

Example

#include "au3Irrlicht2.au3"

_IrrStart(\$IRR_EDT_OPENGL, 800, 600, \$IRR_BITS_PER_PIXEL_32, _

```
$IRR_WINDOWED, $IRR_SHADOWS, $IRR_CAPTURE_EVENTS,
$IRR_VERTICAL_SYNC_ON )
```

```
local $Camera = _IrrAddCamera(10,10,10, 0,0,0 )
local $testNode = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $testNode,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $testNode, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
```

```
_IrrStop()
```

```
Copy to Clipboard
```

_IrrStop

Stops the Irrlicht Engine freeing all of the resources, closing the display window and IrrlichtWrapper.dll.

#Include <au3Irrlicht2.au3>
_IrrStop()

Parameters

None.

Return Value

Success: True Failure: False

Remarks

None.

Related

_IrrStart, _IrrStartAdvanced, _IrrRunning

Example

#include "au3Irrlicht2.au3"

_IrrStart(\$IRR_EDT_OPENGL, 800, 600, \$IRR_BITS_PER_PIXEL_32, _

```
$IRR_WINDOWED, $IRR_SHADOWS, $IRR_CAPTURE_EVENTS,
$IRR_VERTICAL_SYNC_ON )
```

```
local $Camera = _IrrAddCamera(10,10,10, 0,0,0 )
local $testNode = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $testNode,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $testNode, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
```

```
_IrrStop()
```

```
Copy to Clipboard
```

IrrTransparentZWrite

[todo]

#Include <au3Irrlicht2.au3>
_IrrTransparentZWrite()

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetFPS

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetFPS()

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetPrimitivesDrawn

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetPrimitivesDrawn()

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetWindowCaption

Set the caption in the Irrlicht window title bar.

#Include <au3Irrlicht2.au3>
_IrrSetWindowCaption(\$s_Caption)

Parameters

\$s_Caption String for the title bar.

Return Value

Success: True Failure: False

Remarks

None.

Related

Example

#include "au3Irrlicht2.au3"

_IrrStart()

_IrrSetWindowCaption("Title of the window")

```
local $nodeCamera = _IrrAddCamera(10,10,10, 0,0,0 )
local $testNode = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $testNode,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $testNode, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
WHILE _IrrRunning()
_IrrBeginScene(0, 0, 0)
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

Copy to Clipboard

IrrGetScreenSize

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetScreenSize(ByRef \$i_Width, ByRef \$i_Height)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrMaximizeWindow

[todo]

#Include <au3Irrlicht2.au3>
_IrrMaximizeWindow()

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrMinimizeWindow

[todo]

#Include <au3Irrlicht2.au3>
_IrrMinimizeWindow()

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrRestoreWindow

[todo]

#Include <au3Irrlicht2.au3>
_IrrRestoreWindow()

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetResizableWindow

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetResizableWindow(\$i_Resizable)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrMakeARGB

Computes valid 32bit color value including alpha (translucency) as expected from several functions.

#Include <au3Irrlicht2.au3>
_IrrMakeARGB(\$i_Alpha, \$i_Red, \$i_Green, \$i_Blue)

Parameters

\$i_Alpha	Alpha component of the colour.
\$i_Red, \$i_Green, \$i_Blue	Red, green and blue components (0-255).

Return Value

success: 32bit unsigned int colour value including alpha.

Remarks

None.

Related

None.

Example

#include "au3Irrlicht2.au3"

```
_IrrStart()
```

```
local $font = _IrrGetFont( "./media/fonthaettenschweiler.bmp" )
```

```
_IrrAddBillboardTextSceneNode( $font, "au3Irrlicht2 ...", 30.0, 15.0, 0.0, 0.0, 20.0, 0, _
```

```
_IrrMakeARGB(0, 255, 0, 0), _IrrMakeARGB(0, 255,
```

```
255,0))
```

```
_IrrAddBillboardTextSceneNode( $font, "... what else?", 30.0, 15.0, 0.0, -5.0, 17.0, 0, _
```

```
_IrrMakeARGB(0, 255, 255, 0), _IrrMakeARGB(0, 0,
```

```
255, 0))
```

```
_IrrAddFPSCamera()
```

```
WHILE _IrrRunning()
_IrrBeginScene(0,0,25)
_IrrDrawScene()
_IrrEndScene()
WEND
```

```
_IrrStop()
```

```
Copy to Clipboard
```

IrrQueryFeature

[todo]

#Include <au3Irrlicht2.au3>
_IrrQueryFeature(\$i_Feature)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

_IrrGetTime

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetTime()

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

CreateVertStruct

Helper function: creates a structure array of vertices as expected from some functions.

#Include <au3Irrlicht2.au3>
__CreateVertStruct(\$iVert)

Parameters

\$iVert Number of vertex elements the struct array shall contain.

Return Value

Success: The created vertex array struct

Remarks

None

Related

<u>___GetVertStruct</u>, <u>___SetVertStruct</u>, <u>__CreateVectStruct</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()
local \$tVertex = __CreateVertStruct(5)

; set up five vertices to define the points of a pyramid. the vertices have

; many properties that need to be set up to properly define the structure

___SetVertStruct(\$tVertex, 0, \$VERT_X, -20)

___SetVertStruct(\$tVertex, 0, \$VERT_Y, 0)

___SetVertStruct(\$tVertex, 0, \$VERT_Z, -20)

___SetVertStruct(\$tVertex, 1, \$VERT_X, -20)

___SetVertStruct(\$tVertex, 1, \$VERT_Y, 0)

___SetVertStruct(\$tVertex, 1, \$VERT_Z, 20)

___SetVertStruct(\$tVertex, 2, \$VERT_X, 20)

___SetVertStruct(\$tVertex, 2, \$VERT_Y, 0)

___SetVertStruct(\$tVertex, 2, \$VERT_Z, 20)

___SetVertStruct(\$tVertex, 3, \$VERT_X, 20)

___SetVertStruct(\$tVertex, 3, \$VERT_Y, 0)

___SetVertStruct(\$tVertex, 3, \$VERT_Z, -20)

___SetVertStruct(\$tVertex, 4, \$VERT_X, 0)

__SetVertStruct(\$tVertex, 4, \$VERT_Y, 35)

_____SetVertStruct(\$tVertex, 4, \$VERT_Z, 0)

; co-ordinates across a texture run from 0 to 1 so we place each of the vertices ; on this texture plane to appear ; if the pyramid was painted from its bottom up

___SetVertStruct(\$tVertex, 0, \$VERT_TEXTUREX, 0)

- ___SetVertStruct(\$tVertex, 0, \$VERT_TEXTUREY, 0)
- ___SetVertStruct(\$tVertex, 1, \$VERT_TEXTUREX, 0)
- ___SetVertStruct(\$tVertex, 1, \$VERT_TEXTUREY, 1)
- ___SetVertStruct(\$tVertex, 2, \$VERT_TEXTUREX, 1)
- ___SetVertStruct(\$tVertex, 2, \$VERT_TEXTUREY, 1)

___SetVertStruct(\$tVertex, 3, \$VERT_TEXTUREX, 1)

___SetVertStruct(\$tVertex, 3, \$VERT_TEXTUREY, 0)

___SetVertStruct(\$tVertex, 4, \$VERT_TEXTUREX, 0.5)

___SetVertStruct(\$tVertex, 4, \$VERT_TEXTUREY, 0.5)

; each of the vertices can be assigned a colour to tint the texture:

___SetVertStruct(\$tVertex, 0, \$VERT_VCOLOR, _IrrMakeARGB(0, 255, 255, 255))

_SetVertStruct(\$tVertex, 1, \$VERT_VCOLOR, _IrrMakeARGB(0, 255, 255,

```
255))
```

___SetVertStruct(\$tVertex, 2, \$VERT_VCOLOR, _IrrMakeARGB(0, 255, 255, 255))

___SetVertStruct(\$tVertex, 3, \$VERT_VCOLOR, _IrrMakeARGB(0, 255, 255, 255))

___SetVertStruct(\$tVertex, 4, \$VERT_VCOLOR, _IrrMakeARGB(0, 255, 255, 255))

; create the faces, this is an array of indices referencing the vectors they ; are collected into groups of three each defining a triangle in the mesh local \$aIndices[18] = [0,1,4, 1,2,4, 2,3,4, 3,0,4, 2,1,0, 0,3,2]

```
; create the mesh from the array of vertices and indices
local $hMesh = _IrrCreateMesh("testMesh", $tVertex, $aIndices)
```

```
; add the mesh to the scene
```

```
local $SceneNode = _IrrAddMeshToScene( $hMesh )
_IrrAddRotationAnimator($SceneNode, 0, 0.2, 0)
```

```
_IrrSetNodeMaterialTexture( $SceneNode, _IrrGetTexture(
```

```
"./media/au3irr2_logo.jpg" ), 0 )
```

```
_IrrSetNodeMaterialFlag( $SceneNode, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
_IrrAddCamera(30, 30, 30, 0, 10, 0)
```

```
WHILE _IrrRunning()

_IrrBeginScene(0, 0, 25)

_IrrDrawScene()

_IrrEndScene()

WEND
```

_IrrStop()

GetVertStruct

Helper function: Returns a specific value from a structure array of vertices.

#Include <au3Irrlicht2.au3>
___GetVertStruct(ByRef \$tVertex, \$iVertex, \$vMember)

Parameters

\$tVertex	Structure array of vertices as created with CreateVertStruct
\$iVertex	Vertex element from which value shall be returned (0- based!)
\$vMember	One of following values to return: \$VERT_X \$VERT_Y \$VERT_Z \$VERT_NORMALX \$VERT_NORMALY \$VERT_NORMALZ \$VERT_VCOLOR \$VERT_TEXTUREX \$VERT_TEXTUREY

Return Value

Success: Requested \$vMember

Remarks

None

Related

<u>CreateVertStruct</u>, <u>SetVertStruct</u>, <u>CreateVectStruct</u>

Example

```
#include "au3Irrlicht2.au3"
IrrStart()
local $mesh = IrrGetMesh( "./media/cube.x" )
local $tVertex; variable for the vertex array struct
; copy the vertex information into the array
local $vertex_count = _IrrGetMeshVertices( $mesh, 0, $tVertex)
local $i
for $i = 0 to $vertex count - 1; itterate through all of the vertices
  ; shrink vertex X location by half its size, then change vertex colour value
    _SetVertStruct($tVertex, $i, $VERT_X, __GetVertStruct($tVertex, $i,
$VERT_X) * 0.5)
  __SetVertStruct($tVertex, $i, $VERT_VCOLOR, _IrrMakeARGB(0,
Random(0,255), Random(0,255), Random(0,255)))
next; $i
; copy the altered vertex infomation back to the mesh
_IrrSetMeshVertices( $mesh, 0, $tVertex )
; add mesh and camera to the scene:
local $nodeCube = _IrrAddMeshToScene( $mesh )
IrrSetNodeMaterialFlag( $nodeCube, $IRR EMF LIGHTING, $IRR OFF )
IrrSetNodePosition($nodeCube, -0.5, -0.5, 5)
_IrrAddFPSCamera($IRR_NO_OBJECT, 5, 0.01)
```

```
WHILE _IrrRunning()
_IrrBeginScene( 0, 0, 25 )
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

SetVertStruct

Helper function: Sets a value into a structure array of vertices.

#Include <au3Irrlicht2.au3>
___SetVertStruct(ByRef \$tVertex, \$iVertex, \$vMember, \$vData)

Parameters

\$tVertex	Structure array of vertices as created with CreateVertStruct
\$iVertex	Vertex element where specific value shall be set (0-based!)
\$vMember	One of following values to set: \$VERT_X \$VERT_Y \$VERT_Z \$VERT_NORMALX \$VERT_NORMALY \$VERT_NORMALZ \$VERT_VCOLOR \$VERT_TEXTUREX \$VERT_TEXTUREY

Return Value

None.

Remarks

None

Related

<u>CreateVertStruct</u>, <u>GetVertStruct</u>, <u>CreateVectStruct</u>

Example

```
#include "au3Irrlicht2.au3"
```

_IrrStart()

```
local $mesh = _IrrGetMesh( "./media/cube.x" )
```

local \$tVertex; variable for the vertex array struct
; copy the vertex information into the array
local \$vertex_count = _IrrGetMeshVertices(\$mesh, 0, \$tVertex)

local \$i

```
for $i = 0 to $vertex_count - 1 ; itterate through all of the vertices
  ; shrink vertex X location by half its size, then change vertex colour value
```

__SetVertStruct(\$tVertex, \$i, \$VERT_X, __GetVertStruct(\$tVertex, \$i, \$VERT_X) * 0.5)

```
__SetVertStruct($tVertex, $i, $VERT_VCOLOR, _IrrMakeARGB(0, Random(0,255), Random(0,255), Nandom(0,255) ) ) next ; $i
```

```
; copy the altered vertex infomation back to the mesh
_IrrSetMeshVertices( $mesh, 0, $tVertex )
```

```
; add mesh and camera to the scene:
local $nodeCube = _IrrAddMeshToScene( $mesh )
_IrrSetNodeMaterialFlag( $nodeCube, $IRR_EMF_LIGHTING, $IRR_OFF )
IrrSetNodePosition($nodeCube, -0.5, -0.5, 5)
```

```
_IrrAddFPSCamera($IRR_NO_OBJECT, 5, 0.01)
```

```
WHILE _IrrRunning()
```

```
_IrrBeginScene( 0, 0, 25 )
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

CreateVectStruct

Helper function: creates a structure array of vectors as expected from some functions.

#Include <au3Irrlicht2.au3>
__CreateVectStruct(\$iVect)

Parameters

\$iVect	Number of vector elements the created struct array shall contain.
---------	-------------------------------------------------------------------

Return Value

Success: The created vector struct.

Remarks

None

Related

<u>_____SetVectStruct</u>, <u>_____GetVectStruct</u>, <u>___CreateVertStruct</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCamera = _IrrAddCamera( 150,50,0, 0,50,0 )
local $nodeTest = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
local $tVectors = __CreateVectStruct(4)
__SetVectStruct($tVectors, 0, -100, 50, 0)
__SetVectStruct($tVectors, 1, 0, 100, -100)
__SetVectStruct($tVectors, 2, 100, 50, 0)
SetVectStruct($tVectors, 3, 0, 100, 100)
```

```
_IrrAddSplineAnimator( $nodeTest, $tVectors, 0, 0.5, 1)
```

```
WHILE _IrrRunning()
_IrrBeginScene( 0, 0, 25 )
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

GetVectStruct

Helper function: Returns a specific value from a structure array of vectors.

#Include <au3Irrlicht2.au3>
___GetVectStruct(ByRef \$tVector, \$iVector, \$vMember)

Parameters

\$tVector	Structure array of vectors as created with CreateVectStruct
\$iVector	Vector element from which value shall be returned (0- based!)
\$vMember	One of following values to return: \$VECT_X \$VECT_Y \$VECT_Z

Return Value

Success: Requested \$vMember

Remarks

None

Related

<u>CreateVectStruct</u>, <u>SetVectStruct</u>, <u>CreateVertStruct</u>

Example

```
#include "au3Irrlicht2.au3"
```

_IrrStart()

```
local $nodeCamera = _IrrAddCamera( 150,50,0, 0,50,0 )
local $nodeTest = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
local $tVectors = __CreateVectStruct(4)
__SetVectStruct($tVectors, 0, -100, 50, 0)
__SetVectStruct($tVectors, 1, 0, 100, -100)
__SetVectStruct($tVectors, 2, 100, 50, 0)
__SetVectStruct($tVectors, 3, 0, 100, 100)
```

```
_IrrAddSplineAnimator( $nodeTest, $tVectors, 0, 0.5, 1)
```

```
MsgBox (0, "", "Y value of very last vector: " & __GetVectStruct($tVectors, 3, $VECT_Y))
```

```
WHILE _IrrRunning()

_IrrBeginScene( 0, 0, 25 )

_IrrDrawScene()

_IrrEndScene()

WEND
```

_IrrStop()

SetVectStruct

Helper function: Sets x, y, z values into a structure array of vectors.

#Include <au3Irrlicht2.au3>
___SetVectStruct(ByRef \$tVector, \$iVector, \$fX, \$fY, \$fZ)

Parameters

\$tVector	Structure array of vectors as created with CreateVectStruct
\$iVector	Vector element to set (0-based!)
\$fX, \$fY, \$fZ	X, Y, Z values of the vector

Return Value

None.

Remarks

None

Related

<u>CreateVectStruct</u>, <u>GetVectStruct</u>, <u>CreateVertStruct</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeCamera = _IrrAddCamera( 150,50,0, 0,50,0 )
local $nodeTest = _IrrAddTestSceneNode()
_IrrSetNodeMaterialTexture( $nodeTest,
_IrrGetTexture(".\media\au3irr2_logo.jpg"), 0)
_IrrSetNodeMaterialFlag( $nodeTest, $IRR_EMF_LIGHTING, $IRR_OFF )
```

```
local $tVectors = __CreateVectStruct(4)
__SetVectStruct($tVectors, 0, -100, 50, 0)
__SetVectStruct($tVectors, 1, 0, 100, -100)
__SetVectStruct($tVectors, 2, 100, 50, 0)
__SetVectStruct($tVectors, 3, 0, 100, 100)
```

_IrrAddSplineAnimator(\$nodeTest, \$tVectors, 0, 0.5, 1)

```
WHILE _IrrRunning()

_IrrBeginScene( 0, 0, 25 )

_IrrDrawScene()

_IrrEndScene()

WEND
```

_IrrStop()

IrrAddTerrain

Creates a terrain object from a gray scale bitmap.

#Include <au3Irrlicht2.au3>
_IrrAddTerrain(\$s_Path, \$f_PosX = 0.0, \$f_PosY = 0.0, \$f_PosZ = 0.0,
\$f_RotX = 0.0, \$f_RotY = 0.0, \$f_RotZ = 0.0, \$f_ScaleX = 1.0, \$f_ScaleY
= 1.0, \$f_ScaleZ = 1.0, \$i_VertexAlpha = 255, \$i_VertexRed = 255,
\$i_VertexGreen = 255, \$i_VertexBlue = 255, \$i_Smoothing = 0,
\$i_MaxLOD = 5, \$i_PatchSize = \$ETPS_17}

Parameters

\$s_Path	Filename of a gray scale image used to define the contours of the surface
\$f_PosX, \$f_PosY, \$f_PosZ	[optional] Define position of the terrain
\$f_RotX, \$f_RotY, \$f_RotZ	[optional] Define rotation of the terrain
\$f_ScaleX, \$f_ScaleY, \$f_ScaleZ	[optional] Define scaling of the terrain
\$i_VertexAlpha	[optional] Alpha value for the vertex colour from 0 to 255.
\$i_VertexRed, \$i_VertexGreen, \$i_VertexBlue	[optional] Define the vertex colour of all points in the terrain (values from 0 to 255)
\$i_Smoothing	[optional] True or false defines whether the contours of the surface of the terrain are smoothed over.
	[optional] Control the properties of the level of detail

	calculations applied to the terrain. It is recommended that these are left at default values.
\$i_MaxLOD, \$i_PatchSize	 <u>Valid values for \$i_PatchSize:</u> \$ETPS_9 (patch size of 9, at most, use 4 levels of detail with this patch size) \$ETPS_17 (patch size of 17, at most, use 5 levels of detail with this patch size) \$ETPS_33 (patch size of 33, at most, use 6 levels of detail with this patch size) \$ETPS_65 (patch size of 65, at most, use 7 levels of detail with this patch size) \$ETPS_129 (patch size of 129, at most, use 8 levels of detail with this patch size)

Return Value

success: Handle of the terrain object failure: false

Remarks

The terrain is created from a gray scale bitmap where bright pixels are high points on the terrain and black pixels are low points.

You will inevitably have to rescale the terrain during the call or after it is created.

The Terrain object is a special dynamic mesh whose resoloution is reduced in the distance to reduce the number of triangles it consumes.

Related

<u>_IrrScaleTexture</u>

Example

#include "au3Irrlicht2.au3"

_IrrStart()

```
local $nodeTerrain = _IrrAddTerrain( "./media/terrain-heightmap.bmp" )
_IrrSetNodeScale( $nodeTerrain, 40.0, 4.4, 40.0 )
```

_IrrSetNodeMaterialTexture(\$nodeTerrain, _IrrGetTexture("./media/terraintexture.jpg"), 0)

_IrrSetNodeMaterialTexture(\$nodeTerrain, _IrrGetTexture("./media/detailmap3.jpg"), 1)

```
_IrrScaleTexture( $nodeTerrain, 1.0, 60.0 )
```

```
_IrrSetNodeMaterialFlag( $nodeTerrain, $IRR_EMF_LIGHTING, $IRR_OFF
)
IrrSetNodeMaterialType ( $nodeTerrain, $IRR_EMT_DETAIL_MAP )
```

```
local $nodeCamera = _IrrAddFPSCamera()
_IrrSetNodePosition( $nodeCamera, 3942.8, 1102.7, 5113.9 )
```

```
_IrrSetCameraClipDistance( $nodeCamera, 12000 )
```

```
WHILE _IrrRunning()
_IrrBeginScene( 240, 255, 255 )
_IrrDrawScene()
_IrrEndScene()
WEND
```

_IrrStop()

IrrAddTerrainTile

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddTerrainTile(\$h_Image, \$i_TileSize = 256, \$i_DataX = 0, \$i_DataY =
0, \$f_PosX = 0.0, \$f_PosY = 0.0, \$f_PosZ = 0.0, \$f_RotX = 0.0, \$f_RotY =
0.0, \$f_RotZ = 0.0, \$f_ScaleX = 1.0, \$f_ScaleY = 1.0, \$f_ScaleZ = 1.0,
\$i_Smoothing = 1, \$i_MaxLOD = 5, \$i_PatchSize = \$ETPS_17}

Parameters

[param1] [explanation] [moreTextForParam1]

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrAddSphericalTerrain

[todo]

#Include <au3Irrlicht2.au3>
_IrrAddSphericalTerrain(\$s_TopPath, \$s_FrontPath, \$s_BackPath,
\$s_LeftPath, \$RightPath, \$s_BottomPath, \$f_PosX = 0.0, \$f_PosY = 0.0,
\$f_PosZ = 0.0, \$f_RotX = 0.0, \$f_RotY = 0.0, \$f_RotZ = 0.0, \$f_ScaleX =
1.0, \$f_ScaleY = 1.0, \$f_ScaleZ = 1.0, \$i_VertexAlpha = 255, \$i_VertexRed
= 255, \$i_VertexGreen = 255, \$i_VertexBlue = 255, \$i_Smoothing = 0,
\$i_Spherical = 0, \$i_MaxLOD = 5, \$i_PatchSize = \$ETPS_17}

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrGetTerrainHeight

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetTerrainHeight(\$h_Terrain, \$f_X, \$f_Y)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrScaleTexture

Specifies the scaling of a terrain object detail texture

#Include <au3Irrlicht2.au3>
_IrrScaleTexture(\$h_Terrain, \$f_X, \$f_Y)

Parameters

\$h_Terrain Handle of a terrain object
\$f_X, \$f_Y Scaling values for detail texture along x and y axis

Return Value

success: True Failure: False

Remarks

As a terrain object is a particularly huge mesh when textures are applied to it they look extremely pixelated.

To get over this effect a terrain object can have two materials applied to it, one to give general surface color and a second that is copied across the surface like tiles to give a rough detailed texture.

This call specifies the scaling of this detail texture.

Related

<u>_IrrAddTerrain</u>

Example

```
#include "au3Irrlicht2.au3"
```

_IrrStart()

```
local $nodeTerrain = _IrrAddTerrain( "./media/terrain-heightmap.bmp" )
_IrrSetNodeScale( $nodeTerrain, 40.0, 4.4, 40.0 )
```

```
_IrrSetNodeMaterialTexture( $nodeTerrain, _IrrGetTexture( "./media/terrain-texture.jpg" ), 0 )
```

```
_IrrSetNodeMaterialTexture( $nodeTerrain, _IrrGetTexture(
"./media/detailmap3.jpg" ), 1 )
```

```
_IrrScaleTexture( $nodeTerrain, 1.0, 60.0 )
```

```
_IrrSetNodeMaterialType ( $nodeTerrain, $IRR_EMT_DETAIL_MAP )
```

```
local $nodeCamera = _IrrAddFPSCamera()
_IrrSetNodePosition( $nodeCamera, 3942.8, 1102.7, 5113.9 )
```

```
_IrrSetCameraClipDistance( $nodeCamera, 12000 )
```

```
WHILE _IrrRunning()
_IrrBeginScene( 240, 255, 255 )
_IrrDrawScene()
_IrrEndScene()
WEND
```

```
_IrrStop()
```

```
Copy to Clipboard
```

IrrGetTerrainTileHeight

[todo]

#Include <au3Irrlicht2.au3>
_IrrGetTerrainTileHeight(\$h_Terrain, \$f_X, \$f_Y)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrScaleTileTexture

[todo]

#Include <au3Irrlicht2.au3>
_IrrScaleTileTexture(\$h_Terrain, \$f_X, \$f_Y)

Parameters

[param1]	[explanation] [moreTextForParam1]
----------	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrSetTileColor

[todo]

#Include <au3Irrlicht2.au3>
_IrrSetTileColor(\$h_Terrain, \$h_Image, \$i_X=0, \$i_Y=0)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

IrrScaleSphericalTexture

[todo]

#Include <au3Irrlicht2.au3>
_IrrScaleSphericalTexture(\$h_Terrain, \$f_X, \$f_Y)

Parameters

	[explanation] [moreTextForParam1]
--	--------------------------------------

Return Value

[success]: [explanation] [failure]: [explanation] [moreExplanationIndented]

Remarks

[todo]

Related

Credits

<u>Autolt Forum</u> names are mentioned in <brackets> where available.

au3lrr2

au3Irr2 Authors

- J. Rowe <JRowe>
- Andreas Templin <linus>

au3Irr2 Help Files

- One of the above ;)
- Frank Dodd (his FreeBasic Wrapper reference is greatly reusable!)

Additional Contributions

Code and documentation submitted via the forum or by email.

- <ProgAndy>
- <smashly> (also disturbing a sunday morning breakfast coffee is a sin: thanks for excellent work ;))

Last but not least Special Thanks to

- Frank Dodd, his team from the FreeBasic community, and a LOT of others that have supported his Irrlicht Wrapper project with contributions or with technical assistance (see <u>here</u> for detailed acknowledgments).
- Nikolaus Gebhardt and the Irrlicht team.
- All the people that did the used models, textures, ...
- The tool authors around the au3 help saved a lot of time
- All other not-yet-mentioned people sharing time, tips and work for nuts
- Everyone who uses and support au3Irr2



Wrapper Library for Irrlicht

Introduction



Overview

The first thing I must do is acknowledge the fantastic work that has been carried out by the many different authors whos amazing work makes it possible for this package to enable BASIC programmers to build rich and immersive 3D applications and games. You can find a list of acknowledgements at the end of this document.

This wrapper library provides a **simple** and **partial** gateway from non object-oriented languages like **FreeBasic** into the object oriented 3D graphics Engine **Irrlicht**. Although the wrapper only exposes part of the functionality of Irrlicht it endevours to expose the most important parts to give you access to Irrlichts powerful features. It does not directly expose the Irrlicht commands but provides a seperate syntax to encapsulate



the engine. In total it provides well over 200 commands that cover Bitmaps, 3D models, Maj Cameras, Lights and more. At this time there is no practical support for the Irrlic

applications would need to provide any widget controls with.

This document serves as an introduction into the wrapper, from here I suggest yo installation instructions below and then try out the examples, these practical dem provide an excellent way to learn how the wrapper functions. If you need specific your first port of call will be the reference manual and your last port of call the *IrrlichtWrapper.bi* which hopefully you will not need.

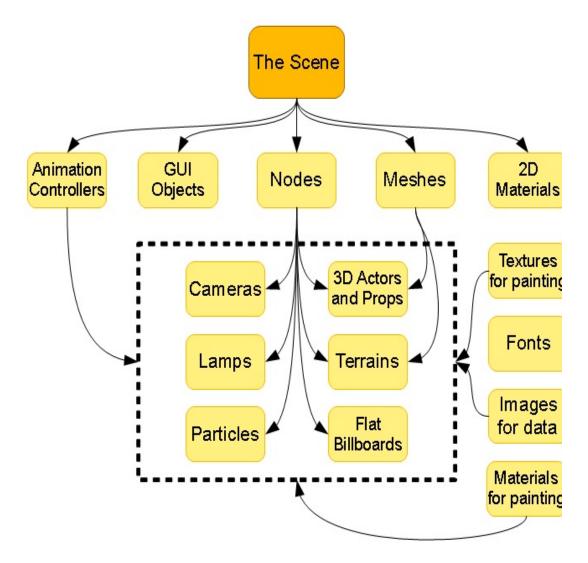
Installation

The installation of the IrrlichtWrapper for FreeBasic package is very simple, just file to a folder of your choice and your ready to go. You may prefer to move the *IrrlichtWrapper.bi* file to your *FreeBasic\inc* directory however I leave it where my project files within the same directory. As the Irrlicht DLL included with this modified to include addition features you cannot use a standard Irrlicht.dll from a download with this package, however if you like to compile your own executable provided to enable you to do this.

Note: You will need FreeBasic verion 0.20 or better to use the wrapper

The Scene

When you start Irrlicht with the *IrrStart* command you create a scene that represent world, this scene is hidden behind the wrapper and is used to load and store all of Bitmap Textures, 3D Meshes, Fonts and the Nodes in your world.



The Texture and Image Objects

The texture and the image objects are both 2D bitmaps that are loaded from a bitmap file.

Textures are used to paint the surface of nodes (3D objects) in the scene or can be drawn directly to the screen as an image for counters or logos. Textures are stored in the computers main memory but, where possible, they are also stored in the Graphics Cards video memory to speed up 3D drawing. When you get a new texture the wrapper will return the object to you as an *irr_texture* type

Images are usualy used as a source of information and are not drawn to the screen and are instead used to set the height of terrains or to describe how grass is distributed. Images are stored only in the computers main memory. When you get a new image file the wrapper will return the object to you as an **irr_image**.

You need to save and manage both of theses objects so you can use them later. Of finished with them it is important that you remove them from memory as images lot of memory and you can easily run out of video memory in particular.

The Font Object

An Irrlicht font is a monochromatic bitmap font that can be used to draw simple text onto the display. When you get a new font the wrapper will return the object to you as an *irr_font* type.

The Mesh Object

The Mesh can be thought of like a blue print and is not actually displayed in your scene. It is a list of triangular geometry that fits together to describe a 3D model, this can be a static prop, a complex map or a character. The mesh can also store animation information and a description of which parts of a texture are to be painted onto its surface.

Where a mesh describes a map it will also have references to dozens of bitmaps that it uses to paint its surfaces.. When you get a new mesh object the wrapper will return it to you as an *irr_mesh* type.

The Node Object

A node is a physical object in your world a 3D Model, Camera, Light, Terrain, Billboard or Particle System. It is an element that has a position, rotation and scale, it will usually be rendered as a visible element if it is in front of the active camera when the canvas is drawn.

Nodes can usually be painted with a texture a process that is referred to as applying a material, they can be moved, rotated and scaled, hidden from view and deleted from the scene.

A 3D model is usually created by adding a mesh object to the scene. When you create a new node it will be returned to you as an *irr_node* type.

The Camera Object

A camera object is a special node type and represent your vantage point into the a camera has a viewpoint and a target at which it looks. Cameras have a series of special commands that can alter the appearance of the display however they can also be into variables of type *irr_node* and be manipulated with all of the node command camera is created it is returned to you an *irr_camera* type.

The Terrain Object

A terrain object is a special node type and consists of a large square mesh that is pulled and lowered to represent hills, valleys and mountains. The terrain can be textured to give the appearance of a realistic landmass and with careful design it can even have map objects buried into it. Terrains have some special commands however they can also be copied directly into variables of type *irr_node* and be manipulated with all of the node commands. When a terrain is created it is returned to you an *irr_texture* type.

The Particle System Object

A particle system is a special node type that can be set up to spit out hundreds of tiny little Billboard like objects and simulate effects like, fire, explosions, fountains, waterfalls, rain etc ... The particle system has an item called an emitters attached to spit out particles and items called affectors attached to fade then out and change their direction. Particle systems have some special commands however they can also be copied directly into variables of type *irr_node* and be manipulated with all of the node commands. When a particle system is created it is returned to you an *irr_particle_system* type.

The Animator Object

Animator objects are attached to nodes and are used to delete, rotate or move the way without you having to control the animation yourself step by step. When you animator it will be returned to you as an *irr_animator* type.

The Selector Object

Selector objects are used to gather together groups of triangular geometry in the s be used for special functions, at the moment the wrapper only supports using the detection. When you create a selector object it will be returned to you as a *irr_se*.

License

The Irrlicht Engine is based in part on the work of the Independent JPEG Group Please refer to the Irrlicht documentation and license agreements for further info

This software is provided 'as-is', without any express or implied warranty. In no ϵ author be held liable for any damages arising from the use of this software. Perm to anyone to use this software for any purpose, including commercial application and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim the original software. If you use this software in a product, an acknowledgment in the documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misre being the original software.

3. This notice may not be removed or altered from any source distribution.

Contact

If you have any problems or suggestions I would be happy to receive them. Alther promise any action or an individual response I do try to help where possible and babies even when I release them into the wild. Mail::frankdodd@ukonline.co.uk

Acknowledgements

I wish to express my gratitude to the following people whose work the wrapper i who have helped with contributions to the wrapper or with technical assistance fc features and problems.

Irrlicht	http://irrlicht.sourceforge.net/author.html		
FreeBasic	http://www.freebasic.net/index.php/about?section=credits		
Simple Terrain	With invaluable help from the OpenGL Splat tutorial by J ϵ		
Spattering	Linden (http://www.cs.auckland.ac.nz/~jvan006/multitex/r		
Grass SceneNode	Released under the Irrlicht License by G Davidson		
Clouds SceneNode	Released under the Irrlicht License by G Davidson		
Lens Flare Scene Node	Placed into the public domain by Paulo Oliveira		
6DOF Camera	With lots of help from Colin MacDonald and theoretical a Aleofjax		
Color and Spline particle affectors	Released under the Irrlicht License by Dark Kilauea		
Newton Physics Engine support	Developed by SiskinEDGE and the library provided by the Dynamics team		
Open Dynamics Engine Physics support	Bindings developed by D.J Peters and the library provided Dynamics Engine team.		
Many Irrilicht commands and support	Provided by Agamemnus and The Car		
FreeType truetype font support	Provided by the FreeType team (www.freetype.org)		
Batching Meshes	Provided by Gaz Davidson (Bitplane)		
Beam Node	Provided by Gaz Davidson (Bitplane)		
Bolt Node	Provided by Sudi and Trivtn under the Irrlicht License		
XEffects - Reloaded	Provided by Blindside		
Many thanks for the development tools	Code::Blocks (www.codeblocks.org); GCC Compiler (gcc FBIde (www.freebasic.net); FBEdit (fbedit.freebasic.net);		
Lots of thanks for	Eponasoft; Daiwa; Crocodudule, John K, AlecZ, Alvaro V		

help in the Forums and everyone that has offered their support. Thanks.

And many thanks to the thousands of mails posts, examples, replies and scattered across the Internet that provided insights into many of the questio to be solved in creating this library.

61	

t-----

Wrapper Library for Irrlicht

Reference Manual



Table of Contents

- <u>Overview</u>
- <u>System</u>
- <u>Keyboard and Mouse</u>
- Filing System
- <u>2D</u>
- <u>Materials</u>
- <u>Scene</u>
- <u>Node</u>
- Animation
- <u>Collision</u>
- <u>Camera</u>
- <u>Lighting</u>
- Particle System
- <u>Terrain</u>
- Graphical User Interface
- <u>Wrapper Structure Definitions</u>

Overview

Before using this manual I suggest that you first read the introduction document that will introduce the basic concepts. While I have made great effort to make this document both comprehensive and accurate there are bound to be errors and omissions please be careful when relying on the information when solving problems.

Irrlicht is a highly impressive and powerful rendering engine that has a wide range of operations that can be used for many types of 3D application, it has an impressive, clean and well designed object oriented structure that is linked to through a simple DLL interface.

Unfortunately its object oriented structure makes it extremely difficult to use from imperative languages like FreeBasic that do not support an object model. This is where the wrapper come in. It provides an interface between languages like FreeBasic and the objected oriented model of Irrlicht. Many functions are translated into simple calls that return the objects back to FreeBasic as simple variables that it can then manage, there is some overhead in going through the wrapper although the interface code has been kept simple to reduce to a minimum.

System

These calls deal with starting, running and stopping the Irrlicht engine, it also includes calls that get system metrics and some other miscellaneous tools.

IrrStart		
Syntax		
IrrStart (device type , screen width as integer, screen height as integer, bits per		
pixel, full screen, use shadows, capture mouse and keyboard, vertical		
syncronisation)		
Description		
Starts the Irrlicht interface and opens a window for rendering.		
device type specifies the renderer to use when drawing to the display this may		
be one of the following types: -		
IRR_EDT_NULL		
A NULL device with no display		
IRR_EDT_SOFTWARE		
Irrlichts default software renderer		
IRR_EDT_SOFTWARE2		
An improved quality software renderer		
IRR_EDT_OPENGL		
Hardware accelerated OpenGL renderer		
IRR_EDT_DIRECT3D8		
Hardware accelerated DirectX 8 renderer (not included in the Wrappers		
'Irlicht.dll' distribution)		
IRR_EDT_DIRECT3D9		
Hardware accelerated DirectX 9 renderer (not included in the Wrappers		
'Irlicht.dll' distribution)		
screen width specifies the width of the viewport in pixels		
corresponded to the bright of the viewport in pixels		

screen height specifies the height of the viewport in pixels

The number of color bits that is used for each pixel 32 bit color gives 24 million different colors whereas 16 bit color gives only 32,000 colors. However the advantage of 16 bit color is that some operations use half the memory and can run at up to twice the speed. This setting can be either of: -

IRR_BITS_PER_PIXEL_16

IRR_BITS_PER_PIXEL_32

Full screen specifies whether the display is to opened in full screen mode or in a window

IRR_WINDOWED

For window mode

IRR_FULLSCREEN

For fullscreen mode. When using full screen mode you will need to adjust the window size to the same dimensions as a supported screen resolution on the target display 640x400 for example.

Use shadows starts the engine in a mode that supports the rendering of stencil shadows.

IRR_NO_SHADOWS

For a display that does not support shadows.

IRR_SHADOWS

For a display that supports shadows.

Capture mouse and keyboard specified whether you want to capture keyboard and mouse events, if you choose to ignore them they will be handled by Irrlicht for FPS camera control. This parameter should be either of: -IRR_IGNORE_EVENTS IRR_CAPTURE_EVENTS

vertical syncronisation specifies whether the display of each new frame is syncronised with vertical refresh of the graphics card. This produces a smoother display and avoids 'tearing' where the viewer can see parts of two different frames at the same time. The setting can be either of :-

IRR_VERTICAL_SYNC_OFF IRR_VERTICAL_SYNC_ON

Example

IrrStart(IRR_EDT_OPENGL, screen_width, screen_height,

IrrStartAdvanced

Syntax

Description

An advanced call for starting the Irrlicht interface and opens a window for rendering.

device type specifies the renderer to use when drawing to the display this may be one of the following types: -

IRR_EDT_NULL A NULL device with no display IRR_EDT_SOFTWARE Irrlichts default software renderer IRR_EDT_SOFTWARE2 An improved quality software renderer IRR_EDT_OPENGL Hardware accelerated OpenGL renderer IRR_EDT_DIRECT3D8

Hardware accelerated DirectX 8 renderer (not included in the Wrappers 'Irlicht.dll' distribution)

IRR_EDT_DIRECT3D9

Hardware accelerated DirectX 9 renderer (not included in the Wrappers 'Irlicht.dll' distribution)

screen width specifies the width of the viewport in pixels screen height specifies the height of the viewport in pixels

The number of color bits that is used for each pixel 32 bit color gives 24 million different colors whereas 16 bit color gives only 32,000 colors. However the advantage of 16 bit color is that some operations use half the memory and can run at up to twice the speed. This setting can be either of: -

IRR_BITS_PER_PIXEL_16 IRR_BITS_PER_PIXEL_32

Full screen specifies whether the display is to opened in full screen mode or in a window

IRR_WINDOWED

For window mode

IRR_FULLSCREEN

For fullscreen mode. When using full screen mode you will need to adjust the window size to the same dimensions as a supported screen resolution on the target display 640x400 for example.

Use shadows starts the engine in a mode that supports the rendering of stencil shadows.

IRR_NO_SHADOWS

For a display that does not support shadows.

IRR_SHADOWS

For a display that supports shadows.

Capture mouse and keyboard specified whether you want to capture keyboard and mouse events, if you choose to ignore them they will be handled by Irrlicht for FPS camera control. This parameter should be either of: -IRR_IGNORE_EVENTS IRR_CAPTURE_EVENTS vertical syncronisation specifies whether the display of each new frame is syncronised with vertical refresh of the graphics card. This produces a smoother display and avoids 'tearing' where the viewer can see parts of two different frames at the same time. The setting can be either of :-

IRR_VERTICAL_SYNC_OFF IRR_VERTICAL_SYNC_ON

devicetype allows a specific type of device for example a windows screen or a console to be selected. For the time being this should be set to 0 which automatically selects the best device

doublebufferenabled is used to control whether double buffering is used. When double buffering is used two drawing surfaces are created one for display and the other that is used for drawing too. Double buffering is required for anitaliasing the options are: IRR_ON or IRR_OFF

antialiasenabled is used to enable the antialiasing effect, this effect produces a blurring at the edges of object giving their lines a smooth natural appearence. There is usually a big penalty for using this effect though sometimes as high as 30% of the frame rate or more. This is a value for the anti-aliasing and should be a power of 2. (e.g: 2, 4, 8, 16)

highprecisionfpu is used to enable high precision Floating point calculations, that produce more accurate result at the expense of a slower operating speed.

Example

-		
IrrStartAdvanced (_		
<pre>IRR_EDT_OPENGL, _</pre>	'	Use OpenGL
512, 512, _	'	in a window 640x480
IRR_BITS_PER_PIXEL_32,	_'	using 32 bit true color
IRR_WINDOWED, _	'	in a window
<pre>IRR_NO_SHADOWS, _</pre>	'	without stencil shadows
<pre>IRR_IGNORE_EVENTS, _</pre>	'	dont capture keystrokes and mouse
IRR_ON, _	'	sync to the monitor refresh rate
0, _	'	0 = use the most appropriate window
device		
IRR_ON, _	'	Switch on double buffering of the
display		
4, _	'	Anti-aliasing level 4
IRR_ON)	'	use high precision floating point
math		

IrrRunning

Syntax IrrRunning

Description Used to determine if the Irrlicht engine is still running.

Example

```
IrrStart( IRR_EDT_OPENGL, screen_width, screen_height,
IRR_WINDOWED, IRR_SHADOWS, IRR_IGNORE_EVENTS )
While IrrRunning
Wend
IrrStop
```

IrrSetViewPort

Syntax

IrrSetViewPort(topX as integer, topY as integer, bottomX as integer, bottomY as integer)

Description

Define the area of the screen into which elements are going to be drawn. This can be used to draw the scene multiple times for split screen effects.

Example

```
IrrSetActiveCamera( FirstCamera )
IrrSetViewPort( 0,0, 200,200 )
IrrDrawScene
```

IrrSetRenderTarget

Syntax

IrrSetRenderTarget (texture As irr_texture, sceneBackgroundColor As Uinteger = 0, clearBackBuffer As Ubyte = 1, clearZBuffer As Ubyte = 1)

Description

Set the target surface for rendering, this allows objects to be rendered to a texture that can then be drawn to the screen or displayed on other objects.

Calling this function with texture set to 0 sets the drawing target back to the screen,.

Texture is a texture created with the special .IrrCreateRenderTargetTexture call. scene background color is generated with the FreeBasic RGBA call and defines the colour used in any clear operation.

clean back buffer when set to IRR_ON erases the background of the texture clear z buffer when set to IRR_ON erases the depth buffer (used by stencil shadows and some shaders)

Example

```
Texture = IrrCreateRenderTargetTexture( 512, 512 )
IrrSetRenderTarget( Texture, RGBA( 0,0,0,0), IRR_ON, IRR_ON )
IrrDrawScene
```

IrrBeginScene

Syntax

IrrBeginScene(Red as integer, Green as integer, Blue as integer)

Description

Starts to draw a frame, erasing the canvas with the specified color. The colors are integer values in the range from 0 (black) to 255 (full intensity)

Example

```
IrrStart( IRR_EDT_OPENGL, screen_width, screen_height,
IRR_WINDOWED, IRR_SHADOWS, IRR_IGNORE_EVENTS )
While IrrRunning
IrrBeginScene( 255, 255, 255 )
IrrDrawScene
IrrEndScene
Wend
IrrStop
```

```
IrrDrawScene
```

Syntax IrrDrawScene

Description

This renders the 3D scene to the canvas, drawing all 3D elements: nodes, particles, billboards, etc

Example IrrStart(IRR_EDT_OPENGL, screen_width, screen_height, IRR_WINDOWED, IRR_SHADOWS, IRR_IGNORE_EVENTS) While IrrRunning IrrBeginScene(255, 255, 255) IrrDrawScene IrrEndScene Wend IrrStop

IrrDrawSceneToTexture

Syntax

IrrDrawSceneToTexture(render_texture as irr_texture)

Description

Draw scene manager objects to a texture surface, the texture must have been created with a call to IrrCreateRenderTargetTexture. This is useful for creating textures from 3D objects in your scene perhaps nameplates in the interface for characters for example. NoteThe target texture must be *smaller* than the view window as some resources are shared between the two.

Example

```
IrrSetActiveCamera ( StaticCamera )
IrrDrawSceneToTexture ( RenderTexture )
```

```
IrrBeginScene( 240, 255, 255 )
```

```
IrrSetActiveCamera ( FPSCamera )
IrrDrawScene
```

IrrDrawGUI

Syntax IrrDrawGUI

Description This renders the 2D graphical user interface that has been created to the scene. At the moment this wrapper only supports a static text object for experimentation purposes only.

Example

IrrStart(IRR_EDT_OPENGL, screen_width, screen_height, IRR_WINDOWED, IRR_SHADOWS, IRR_IGNORE_EVENTS) While IrrRunning IrrBeginScene(255, 255, 255) IrrDrawScene IrrDrawGUI IrrEndScene Wend IrrStop

IrrEndScene

Syntax IrrEndScene

Description

This renders the 3D scene to the canvas, drawing all 3D elements: nodes, particles, billboards, etc

Example

```
IrrStart( IRR_EDT_OPENGL, screen_width, screen_height,
IRR_WINDOWED, IRR_SHADOWS, IRR_IGNORE_EVENTS )
While IrrRunning
IrrBeginScene( 255, 255, 255 )
IrrDrawScene
IrrEndScene
Wend
IrrStop
```

<mark>IrrStop</mark> Syntax

IrrStop

Description Stop the Irrlicht Engine freeing all of the resources and closing the display window. Example

IrrStart(IRR_EDT_OPENGL, screen_width, screen_height, IRR_WINDOWED, IRR_SHADOWS, IRR_IGNORE_EVENTS) While IrrRunning IrrBeginScene(255, 255, 255) IrrDrawScene IrrEndScene Wend IrrStop

IrrTransparentZWrite

Syntax IrrTransparentZWrite

Description

Allow transparency to write to the z buffer, this is nessecary sometimes to correct problems with the ordering of transparent objects in the scene, it may also have an effect of performance however.

Example

IrrTransparentZWrite

IrrGetFPS

Syntax

Integer_variable = IrrGetFPS

Description

Get the current frame rate. This is determined by the number of times the IrrEndScene is called per second.

Example

```
frame_rate = IrrGetFPS
IrrStop()
Print "Frame Rate was ";frame_rate
Sleep
```

IrrGetScreenSize

Syntax IrrGetScreenSize(width as integer, height as integer)

Description

Gets the screen side into the two supplied variables.

Example

IrrGetScreenSize(ScreenWidth, ScreenHeight)

IrrGetPrimitivesDrawn

Syntax

unsigned_Integer_variable = IrrGetPrimitivesDrawn

Description

Get the current frame rate. This is determined by the number of times the IrrEndScene is called per second.

Example

polygons = IrrGetPrimitivesDrawn IrrStop() Print "The system drew about ";polygons;" triangles" Sleep

IrrSetWindowCaption

Syntax

IrrSetWindowCaption(caption text as wide string)

Description Set the caption in the Irrlicht window title bar..

Example

IrrSetWindowCaption("Irrlicht in Free Basic")

IrrMakeARGB

Syntax

unsigned_integer = IrrMakeARGB (Alpha, Red, Green, Blue)

Description

Takes four values representing a colors Alpha, Red, Green and Blue intensity and returns them as a 32bit unsigned integer. Typically used for working with colors in IRR_VECT structures.

Example

vcolor = IrrMakeARGB(0, 255, 128, 128)

IrrQueryFeature

Syntax

uinteger IrrQueryFeature(Feature as IRR_VIDEO_FEATURE_QUERY)

Description

Used to determine if a particular video feature is supported by the graphics card. The function will return (1) if the feature is supported and (0) if it isnt. The feature parameter should be either of the following values: -

EVDF_RENDER_TO_TARGET

Is driver able to render to a surface?

EVDF_HARDWARE_TL

Is hardeware transform and lighting supported?

EVDF_MULTITEXTURE

Are multiple textures per material possible?

EVDF_BILINEAR_FILTER

Is driver able to render with a bilinear filter applied?

EVDF_MIP_MAP

Can the driver handle mip maps?

EVDF_MIP_MAP_AUTO_UPDATE

Can the driver update mip maps automatically?

EVDF_STENCIL_BUFFER

Are stencilbuffers switched on and does the device support stencil buffers?

EVDF_VERTEX_SHADER_1_1

Is Vertex Shader 1.1 supported?

EVDF_VERTEX_SHADER_2_0

Is Vertex Shader 2.0 supported?

EVDF_VERTEX_SHADER_3_0

Is Vertex Shader 3.0 supported?

EVDF PIXEL SHADER 1 1 Is Pixel Shader 1.1 supported? EVDF_PIXEL_SHADER_1_2 Is Pixel Shader 1.2 supported? EVDF_PIXEL_SHADER_1_3 Is Pixel Shader 1.3 supported? EVDF_PIXEL_SHADER_1_4 Is Pixel Shader 1.4 supported? EVDF_PIXEL_SHADER_2_0 Is Pixel Shader 2.0 supported? EVDF_PIXEL_SHADER_3_0 Is Pixel Shader 3.0 supported? EVDF_ARB_VERTEX_PROGRAM_1 Are ARB vertex programs v1.0 supported? EVDF ARB FRAGMENT PROGRAM 1 Are ARB fragment programs v1.0 supported? EVDF_ARB_GLSL Is GLSL supported? EVDF HLSL Is HLSL supported? EVDF_TEXTURE_NPOT Are non-power-of-two textures supported? EVDF_FRAMEBUFFER_OBJECT Are framebuffer objects supported? EVDF_VERTEX_BUFFER_OBJECT Are vertex buffer objects supported? EVDF ALPHA TO COVERAGE Is alpha to coverage supported? EVDF_COLOR_MASK Are color masks supported? EVDF MULTIPLE RENDER TARGETS Are multiple render targets supported? EVDF MRT BLEND Are seperate blend settings for render targets supported? EVDF_MRT_COLOR_MASK Are seperate color masks for render targets supported? EVDF MRT BLEND FUNC Are seperate blend functions for render targets supported? EVDF GEOMETRY SHADER

Are geometry shaders supported?

Example

IrrDisableFeature

Syntax

uinteger IrrDisableFeature(Feature as IRR_VIDEO_FEATURE_QUERY, state as uinteger)

Description

Used to disable a particular video feature on the graphics card. The feature parameter is identical to IrrQueryFeature.

State should be either IRR_ON or IRR_OFF

Example

IrrDisableFeature(EVDF_MULTITEXTURE, IRR_OFF)

IrrGetTime

Syntax unsigned_integer = IrrGetTime

Description Get the current time in milliseconds.

Example

time = IrrGetTime

IrrSetTime

Syntax IrrGetTime(time as uinteger)

Description

Set the current animation time in milliseconds.

Example

IrrSetTime(2500)

IrrIsFullscreen

Syntax IrrIsFullscreen() as integer

Description

Checks if the Irrlicht window is running in fullscreen mode. Returns 0 if the application is windowed any other value indicates full screen mode

Example

if IrrIsFullscreen = IRR_OFF Then Print "Windowed Mode"

IrrIsWindowActive

Syntax IrrIsWindowActive() as integer

Description Checks if Irrlicht window is active. Returns 0 if the application is windowed any other value indicates full screen mode

Example

if IrrIsWindowActive > 0 Then Print IrrDrawScene

IrrIsWindowFocused

Syntax IrrIsWindowFocused() as integer

Description Checks if the Irrlicht window has focus. Returns 0 if the application is windowed any other value indicates full screen mode Example

if IrrIsWindowFocused > 0 Then Print IrrDrawScene

IrrIsWindowMinimized

Syntax

IrrIsWindowMinimized() as integer

Description

Checks if the Irrlicht window is minimized. Returns 0 if the application is windowed any other value indicates full screen mode

Example

if IrrIsWindowMinimized = 0 Then Print IrrDrawScene

IrrMaximizeWindow

Syntax IrrMaximizeWindow()

Description Maximizes the window if possible.

Example

IrrMaximizeWindow

IrrMinimizeWindow

Syntax IrrMinimizeWindow()

Description Minimizes the window if possible.

Example

IrrMinimizeWindow

IrrRestoreWindow

Syntax IrrRestoreWindow()

Description

Restore the window to normal size if possible.

Example

_____ IrrRestoreWindow

IrrResizableWindow
Syntax
IrrResizableWindow()
Description
Make the irrlicht window resizable by dragging on the corner of the window.
Example
IrrResizableWindow

Keyboard and Mouse

These calls allow you recover keyboard events and mouse actions that the user creates.

IrrKeyEventAvailable

Syntax IrrSetWindowCaption

Description Determine if there are any keystrokes waiting to be read..

Example

while IrrKeyEventAvailable KeyEvent = IrrReadKeyEvent Wend

IrrReadKeyEvent

Syntax

irr_key_event_pointer = IrrReadKeyEvent

Description

Read a key event from the Irrlicht window the properties of the key event are stored in the returned type.

Example

```
While IrrKeyEventAvailable
KeyEvent = IrrReadKeyEvent
If KeyEvent->key = IRR_KEY_DOWN then
Movement = DOWN
End If
Wend
```

IrrMouseEventAvailable

Syntax IrrMouseEventAvailable

Description Determine if there are any mouse actions waiting to be read.

Example

```
while IrrMouseEventAvailable
MouseEvent = IrrReadMouseEvent
Wend
```

IrrReadMouseEvent

Syntax

irr_mouse_event_pointer = IrrReadMouseEvent

Description

Read a mouse event from the Irrlicht window the properties of the mouse event are stored in the returned type.

Example

```
while IrrMouseEventAvailable
    ' read the mouse event out
    MouseEvent = IrrReadMouseEvent
    if MouseEvent->action = IRR_EMIE_MOUSE_MOVED then
        SPIN = MouseEvent->x
    endif
wend
```

IrrSetMousePosition

Syntax

IrrSetMousePosition(x as single, y as single)

Description

Set the position of the mouse pointer and return the relative change in position.

Example

IrrSetMousePosition(XPosition, YPosition)

IrrGetAbsoluteMousePosition

Syntax

IrrGetAbsoluteMousePosition(x as single, y as single)

Description Gets the absoloute position of the mouse pointer.

Example

IrrGetAbsoluteMousePosition(XPosition, YPosition)

IrrHideMouse	
Syntax IrrHideMouse	
Description Hide the mouse pointer	
Example	
IrrHideMouse	

IrrShowMouse

Syntax IrrShowMouse

Description Shows the mouse pointer

Example

IrrShowMouse

IrrDisplayMouse

Syntax IrrDisplayMouse(hide or show the mouse as integer) Description Hide or show the mouse pointer while it is within the Irrlicht display. There are two macro's available for the function IrrHideMouse and IrrShowMouse to simply hide or show the mouse. 1 shows the mouse pointer and 0 hides it.

Example

IrrDisplayMouse(0)

Filing System

These calls deal with the way irrlicht operates with the filing system and adds archives to its a virtual filling system allowing you to compress data into zipfiles that you can access without decompressing them.

IrrAddZipFile
Syntax
IrrAddZipFile(zip file as zstring, ignore case, ignore paths)
Description
Adds a zip archive to the filing system allowing you to load files straight out of
the zip file. Common pk3 files are simply zip files
Ignore case should be one of the following values: -
IRR_USE_CASE
IRR_IGNORE_CASE
Ignore paths allows you to simply use the filename without the path, the
filename should always be unique in the archive when using this option. The
value should be one of the following: -
IRR_USE_PATHS
IRR_IGNORE_PATHS
Example
<pre>IrrAddZipFile("data.pk3", IRR_IGNORE_CASE, IRR_IGNORE_PATHS)</pre>

IrrChangeWorkingDirectory

Syntax

IrrChangeWorkingDirectory(New directory as zstring)

Description Change the working directory of the Irrlicht Environment.

Example

IrrChangeWorkingDirectory("c:\media")

IrrGetWorkingDirectory

Syntax

string = IrrGetWorkingDirectory

Description

Get the current working directory of the Irrlicht Environment.

Example

CurrentDirectory = IrrGetWorkingDirectory

2**D**

Support for 2D operations including the loading of bitmaps that can be used for texturing 3D objects or for display on the screen as 2D sprites.

IrrSetTextureCreationFlag

Syntax

IrrSetTextureCreationFlag(flag_to_set as IRR_TEXTURE_CREATION_FLAG, flag_value as uinteger)

Description

Sets texture creation flags controlling how textures are handled when they are created. The following flags can be set: -

ETCF_ALWAYS_16_BIT

Forces the driver to always create 16 bit textures, independently of which format the file on disk has. When choosing this you may loose some color detail, but gain speed and save memory. 16 bit textures can be transferred twice as quickly as 32 bit textures and only use half of the memory space. When using this flag, it does not make sense to use the flags ETCF_ALWAYS_32_BIT, ETCF_OPTIMIZED_FOR_QUALITY, or ETCF_OPTIMIZED_FOR_SPEED at the same time.

ETCF_ALWAYS_32_BIT

Forces the driver to always create 32 bit textures, independently of which format the file on disk has. Please note that some drivers (like the software device) will ignore this, because they are only able to create and use 16 bit textures. When using this flag, it does not make sense to use the flags ETCF_ALWAYS_16_BIT, ETCF_OPTIMIZED_FOR_QUALITY, or ETCF_OPTIMIZED_FOR_SPEED at the same time.

ETCF_OPTIMIZED_FOR_QUALITY

Lets the driver decide in which format the textures are created and tries to make

the textures look as good as possible. Usually it simply chooses the format in which the texture was stored on disk. When using this flag, it does not make sense to use the flags ETCF_ALWAYS_16_BIT, ETCF_ALWAYS_32_BIT, or ETCF_OPTIMIZED_FOR_SPEED at the same time.

ETCF_OPTIMIZED_FOR_SPEED

Lets the driver decide in which format the textures are created and tries to create them maximizing render speed. When using this flag, it does not make sense to use the flags ETCF_ALWAYS_16_BIT, ETCF_ALWAYS_32_BIT, or ETCF_OPTIMIZED_FOR_QUALITY, at the same time.

ETCF_CREATE_MIP_MAPS Automatically creates mip map levels for the textures.

ETCF_NO_ALPHA_CHANNEL Discard any alpha layer and use non-alpha color format.

Example

IrrSetTextureCreationFlag(ETCF_ALWAYS_32_BIT, IRR_ON)

IrrGetTexture

Syntax

irr_texture = IrrGetTexture(Texture file name as zstring)

Description

Load a 2D texture from a bitmap file into video memoy that can then be used to texture a model or to draw onto the screen.

Example

IrrlichtLogo = IrrGetTexture("irrlicht.bmp")

IrrGetImage

Syntax

irr_texture = IrrGetImage(Texture file name as zstring)

Description

Load a 2D texture from a bitmap file into main memory that can then be used to supply a heightmap to a terrain or other similar CPU based operations. The images can not be used to texture 3D objects.

Example

TerrainMap = IrrGetImage("heightmap.bmp")

IrrCreateTexture

Syntax

irr_texture = IrrCreateTexture(texture_name as zstring, x_size as integer, y_size as integer, format as IRR_COLOR_FORMAT)

Description

Creates a blank texture. The format of the texture can be one of the following: -

ECF_A1R5G5B5

16 bit color format used by the software driver, and thus preferred by all other irrlicht engine video drivers. There are 5 bits for every color component, and a single bit is left for alpha information.

ECF_R5G6B5 Standard 16 bit color format.

ECF_R8G8B8

24 bit color, no alpha channel, but 8 bit for red, green and blue.

ECF_A8R8G8B8

Default 32 bit color format. 8 bits are used for every component: red, green, blue and alpha.

Example

```
IrrlichtLogo = IrrCreateTexture( "mytexture", 128, 128,
ECF_A8R8G8B8 )
```

IrrCreateImage

Syntax

irr_image = IrrCreateImage(x_size as integer, y_size as integer, format as IRR_COLOR_FORMAT)

Description

Creates a blank image, does not use video memory. The format of the image can be one of the following: -

ECF_A1R5G5B5

16 bit color format used by the software driver, and thus preferred by all other irrlicht engine video drivers. There are 5 bits for every color component, and a single bit is left for alpha information.

ECF_R5G6B5 Standard 16 bit color format.

ECF_R8G8B8 24 bit color, no alpha channel, but 8 bit for red, green and blue.

ECF_A8R8G8B8

Default 32 bit color format. 8 bits are used for every component: red, green, blue and alpha.

Example

BlankPicture = IrrCreateImage(128, 128, ECF_A8R8G8B8)

IrrRemoveTexture

Syntax IrrRemoveTexture(texture as irr_texture)

Description

Removes the texture from memory freeing up the space it occupied. You should ensure that the texture is not in use by materials assigned to nodes.

Example

DIM MyTexture as irr_texture

```
MyTexture = IrrGetTexture( "irrlicht.bmp" )
IrrRemoveTexture( MyTexture )
```

IrrRemoveImage

Syntax

IrrRemoveImage(image as irr_image)

Description

Removes the image from memory freeing up the space it occupied. You should ensure that the image is not in use by other functions.

Example

```
DIM MyImage as irr_image
MyImage = IrrGetImage( "irrlicht.bmp" )
IrrRemoveImage( MyImage )
```

IrrLockTexture

Syntax

pixels_ptr = IrrLockTexture(texture as irr_texture)

Description Locks the texture and returns a pointer to the pixels.

Example

```
DIM texture_pixels as uinteger ptr
texture_pixels = IrrLockTexture( MyTexture )
```

IrrUnlockTexture

Syntax

IrrUnlockTexture(texture as irr_texture)

Description Unlock the texture, presumably after it has been modified and recreate the mipmap levels.

IrrUnlockTexture(MyTexture)

IrrLockImage

Syntax

pixels_ptr = IrrLockImage(image as irr_image)

Description Locks the image and returns a pointer to the pixels.

Example

```
DIM image_pixels as uinteger ptr
image_pixels = IrrLockImage( MyImage )
```

IrrUnlockImage

Syntax IrrUnlockImage(image as irr_image)

Description Unlock the image, presumably after it has been modified.

Example

IrrUnlockImage(MyImage)

IrrCreateRenderTargetTexture

Syntax

irr_texture = IrrCreateRenderTargetTexture(x_size as integer, y_size as integer)

Description

Create a texture that is suitable for the scene manager to use as a surface to which it can render its 3d object. Each of the dimentions must be of a power of two for example 128x128 or 256x256.

This function is very important when producing texture maps for special effects for example a rendering of a model for a 2D image displayed in the HUD, the rendering of a model for display on a 3D surface for example a video display of

virtual camera, the rendering of the texture for the reflection of a mirror, the rendering of the environment for use in a water or chrome shader. Most cards, even old cards, will support this very important function.

Example

RenderTexture = IrrCreateRenderTargetTexture (256, 256)

IrrMakeNormalMapTexture

Syntax

IrrMakeNormalMapTexture(Texture object as irr_texture, Amplitude as single)

Description

Create a normal map from a gray-scale height map texture. Normal maps are used to add a high level of surface lighting detail to what are normally low resolution models. They can have a massive effect on the realism of an object, the model you create will have to be created in "tangent" space to support this.

Example

IrrMakeNormalMapTexture(WallBumps, 0.9

IrrColorKeyTexture

Syntax

IrrColorKeyTexture(Texture object as irr_texture, Red as integer, Green as integer, Blue as integer)

Description

Copies any parts of the texture that are the same as the specified color into the textures alpha channel. This can then be used for special effects or to make these regions transparent.

Example

IrrColorKeyTexture(IrrlichtLogo, 255, 255, 255)

IrrDraw2DImage

Syntax

IrrDraw2DImage(Texture to draw as irr_texture, X position as integer, Y position as integer)

Description

Draws the texture to the display at the supplied co-ordinates.

Example

IrrDraw2DImage(IrrlichtLogo, 4, 4)

IrrDraw2DImageElement

Syntax

IrrDraw2DImageElement(Texture to draw as irr_texture, X position as integer, Y position as integer, Source top X as integer, Source top Y as integer, Source bottom X as integer, Source bottom Y as integer, whether to use alpha as integer)

Description

Draws the texture to the display at the supplied co-ordinates, the image is copied from the specified rectangle in the source texture, this enables you to put many images onto a single texture. This function also supports the alpha channel when drawing the image to the display and can draw the image transparently.

The value for whether or not to use the alpha channel should be one of the following values: -

IRR_IGNORE_ALPHA IRR_USE_ALPHA

Example

```
IrrDraw2DImageElement( FreeBasicLogo, screen_width - 60 - 4,
4,0,0,60,31, IRR_USE_ALPHA )
```

IrrDraw2DImageElementStretch

Syntax

IrrDraw2DImageElementStretch (texture as irr_texture, destination top X as integer, destination top Yas integer, destination bottom X as integer,

destination bottom Y as integer, source top X as integer, source top Y as integer, source bottom X as integer, source bottom Y as integer, use Alpha as integer)

Description

Draws the texture to the display into the supplied rectangle, the image is copied from the specified rectangle in the source texture, this enables you to put many images onto a single texture. If the rectangles are different sizes this function will scale the images appropriately. This function also supports the alpha channel when drawing the image to the display and can draw the image transparently.

The value for whether or not to use the alpha channel should be one of the following values: -IRR_IGNORE_ALPHA

IRR_USE_ALPHA

Example

IrrDraw2DImageElementStretch(FreeBasicLogo, 16, 16, 80, 80, 0, 0, 0, 32, 32, IRR_USE_ALPHA)

IrrGetFont

Syntax

irr_font = IrrGetFont(Filename of the bitmap font file as zstring)

Description Loads a bitmap containing a bitmap font.

Example

BitmapFont = IrrGetFont ("bitmapfont.bmp")

Irr2DFontDraw

Syntax

Irr2DFontDraw (Font Object as irr_texture, The text to display as wstring ptr, Top X as integer, Top Y as integer, Bottom X as integer, Bottom Y as integer)

Description

Draws the text into the supplied rectangular area using the supplied font object.

Example

```
Irr2DFontDraw ( BitmapFont, "SIMPLE MONOCHROME FONT", 120, 80,
250, 96 )
```

IrrSaveScreenShot

Syntax

IrrSaveScreenShot(filename as zstring)

Description

Save a screenshot out to a file, the image format is defined by the extension applied to the filename. Irrlicht currently supports: bmp, png, tga, ppm and jpg

Example

IrrSaveScreenShot("c:\myscreen.bmp")

IrrGetScreenShot

Syntax

texture = IrrGetScreenShot(x as uinteger, y as uinteger, width as uinteger, height as uinteger)

Description

Return a pointer to a texture containing a rectangular portion of a screenshot.

Example

DIM texture as irr_texture = IrrGetScreenShot(0,0, 256,256)

IrrGetTextureInformation

Syntax

texture = IrrGetTextureInformation (texture as irr_texture, textureWidth as unsigned integer, textureHeight as unsigned integer, texturePitch as unsigned integer,textureFormat as IRR_COLOR_FORMAT)

Description

Get information on a texture. The width, height, pitch and color format is returned in the supplied variables.

Example

```
IrrGetTextureInformation ( selectedTexture, width, height, pitch,
col_format )
```

IrrGetImageInformation

Syntax

texture = IrrGetImageInformation (image as irr_image, textureWidth as unsigned integer, textureHeight as unsigned integer, texturePitch as unsigned integer,textureFormat as IRR_COLOR_FORMAT)

Description

Get information on an image. The width, height, pitch and color format is returned in the supplied variables.

Example

IrrGetImageInformation (selectedImage, width, height, pitch, col_format)

Materials

Calls for creating and manipulating materials that can be applied to a node to color and texture the object. Basic Materials set common properties like the shininess and reflective color of the objects. Advanced Materials use GPU programs to create sophisticated texturing effects that can greatly add to the realism of the scene but are only supported by modern graphics cards with Pixel and Vertex shader support. Currently Irrlicht supports Vertex Shaders, Pixel Shaders, ARB Vertex programs, ARB Fragment programs, HLSL (DirectX 9) and GLSL (OpenGL).

IrrSetNodeAmbientColor

Syntax

IrrSetNodeAmbientColor (node As irr_node, uColor As Uinteger)

Description

Sets the ambient color of all materials in a node. This color value is created with the FreeBasic RGBA call. The ambient color is a color applied to the whole node as a simulation of ambient lighting reflected from the objects around it.

Example

IrrSetNodeAmbientColor (object_material, RGBA(128,0,0,0))

IrrSetNodeDiffuseColor

Syntax

IrrSetNodeDiffuseColor (node As irr_node, uColor As Uinteger)

Description

Sets the diffuse color of all materials in a node. This color value is created with the FreeBasic RGBA call. The diffuse color is the indirectly lit surface colour.

Example

IrrSetNodeDiffuseColor (object_material, RGBA(128,0,0,0))

IrrSetNodeSpecularColor

Syntax

IrrSetNodeSpecularColor (node As irr_node, uColor As Uinteger)

Description

Sets the specilar color of all materials in a node. This color value is created with the FreeBasic RGBA call. The specular color is the color of the highlights on the node representing reflections of light sources.

Example

IrrSetNodeSpecularColor (object_material, RGBA(128,0,0,0))

IrrSetNodeEmissiveColor

Syntax

IrrSetNodeEmissiveColor (node As irr_node, uColor As Uinteger)

Description

Sets the emissive color of all materials in a node. This color value is created with the FreeBasic RGBA call. The emissive colour is the light 'generated within' the node. Setting this to 255,255,255,255 will make the node appear as though it has the no lighting effect applied to it.

Example

IrrSetNodeEmissiveColor (object_material, RGBA(128,0,0,0))

IrrSetNodeColorByVertex

Syntax

IrrSetNodeColorByVertex (material as irr_material, affected_property as IRR_COLOR_MATERIAL)

Description

Sets which aspect of all of the materials in a node is affected by the vertex colour.

affected_property can be one of: -

ECM_NONE Dont use vertex color for lighting

ECM_DIFFUSE Use vertex color for diffuse light, (the default value)

ECM_AMBIENT Use vertex color for ambient light

ECM_EMISSIVE Use vertex color for emissive light

ECM_SPECULAR Use vertex color for specular light

ECM_DIFFUSE_AND_AMBIENT Use vertex color for both diffuse and ambient light

Example

IrrSetNodeColorByVertex (object_material, ECM_NONE)

IrrMaterialVertexColorAffects

Syntax

IrrMaterialVertexColorAffects (material as irr_material, affected_property as IRR_COLOR_MATERIAL)

Description

Sets which aspect of the material is affected by the vertex colour.

affected_property can be one of: -

ECM_NONE Dont use vertex color for lighting

ECM_DIFFUSE Use vertex color for diffuse light, (the default value)

ECM_AMBIENT Use vertex color for ambient light

ECM_EMISSIVE Use vertex color for emissive light

ECM_SPECULAR Use vertex color for specular light

ECM_DIFFUSE_AND_AMBIENT Use vertex color for both diffuse and ambient light

Example

IrrSetMaterialBlend

Syntax

IrrSetMaterialBlend (material as irr_material, source as IRR_BLEND_FACTOR, destination as IRR_BLEND_FACTOR)

Description

Sets the source and destination surface blend factors for the ONETEXTURE_BLEND material. This is mainly useful in multi-pass rendering, where you render the scene to the display and then render the scene a second time with the ONETEXTURE_BLEND material setting which mixes the existing pixels and the new pixels using the blend setting defined here. IRR_BLEND_FACTOR can be one of the following values: -

EBF_ZERO A fixed value of zero

EBF_ONE A fixed value of one

EBF_DST_COLOR The destination color

EBF_ONE_MINUS_DST_COLOR The inverted destination color

EBF_SRC_COLOR The source color

EBF_ONE_MINUS_SRC_COLOR The inverted source color

EBF_SRC_ALPHA The source alpha value

EBF_ONE_MINUS_SRC_ALPHA The inverted source alpha value

EBF_DST_ALPHA The destination alpha value

EBF_ONE_MINUS_DST_ALPHA The inverted destination alpha value

EBF_SRC_ALPHA_SATURATE

```
IrrSetMaterialBlend ( object_material, EBF_SOURCE_COLOR,
EFB_DST_COLOR )
```

IrrMaterialSetShininess

Syntax

IrrMaterialSetShininess (material as irr_material, shininess as single)

Description

Set how shiny the material is, the higher the value the more defined the highlights.

Example

IrrMaterialSetShininess (object_material, 20.0)

IrrMaterialSetSpecularColor

Syntax

IrrMaterialSetSpecularColor (material as irr_material, Alpha as uinteger, Red as uinteger, Green as uinteger, Blue as uinteger)

Description

Set the color of specular highlights on objects with this material applied.

Example

IrrMaterialSetSpecularColor (object_material, 0, 255, 128, 128

IrrMaterialSetDiffuseColor

Syntax

IrrMaterialSetDiffuseColor (material as irr_material, Alpha as uinteger, Red as uinteger, Green as uinteger, Blue as uinteger)

Description Set the color of diffuse lighting on objects with this material applied.

Example

IrrMaterialSetDiffuseColor (object_material, 0, 255, 128, 255)

IrrMaterialSetAmbientColor

Syntax

IrrMaterialSetAmbientColor (material as irr_material, Alpha as uinteger, Red as uinteger, Green as uinteger, Blue as uinteger)

Description

Set the color of ambient light reflected by objects with this material applied.

Example

IrrMaterialSetEmissiveColor

Syntax

IrrMaterialSetEmissiveColor (material as irr_material, Alpha as uinteger, Red as uinteger, Green as uinteger, Blue as uinteger)

Description

Set the color of light emitted by objects with this material applied.

Example

IrrMaterialSetEmissiveColor (object_material, 0, 64, 128, 255)

IrrMaterialSetMaterialTypeParam

Syntax

IrrMaterialSetMaterialTypeParam(material as irr_material, value as single)

Description

Set material specific parameter. Used in a couple of vertex alpha and normal mapping material types.

Example

IrrSetMaterialLineThickness

Syntax

IrrSetMaterialLineThickness(material as irr_material, thickness as single)

Description

Sets the line thickness of none 3D elements associated with this material.

Example

IrrSetMaterialLineThickness(object_material, 2.0)

IrrAddHighLevelShaderMaterial

Syntax

irr_shader = IrrAddHighLevelShaderMaterial (vertex_program as zstring ptr, vertex_start_function as zstring ptr, vertex_prog_type as uinteger, pixel_program as zstring ptr, pixel_start_function as zstring ptr, pixel_prog_type as uinteger, material_type as uinteger)

Description

Creates a new material using a high level shading language.

vertex program: String containing the source of the vertex shader program. This can be 0 if no vertex program shall be used.

vertex_start_function: Name of the entry function of the vertex shader program vertex_program_type: Vertex shader version used to compile the GPU program pixel_program: String containing the source of the pixel shader program. This can be 0 if no pixel shader shall be used.

pixel_start_function: Entry name of the function of the pixel shader program pixel_program_type: Pixel shader version used to compile the GPU program baseMaterial: Base material which renderstates will be used to shade the material.

Returns a type that contains a material_type number that can be used to shade nodes with this new material. If the shader could not be created it will return 0

Example

shader = IrrAddHighLevelShaderMaterial (_
 vertex_program, "main", IRR_EVST_VS_1_1, _
 pixel_program, "main", IRR_EPST_PS_1_1, _
 IRR_EMT_SOLID)

${\bf IrrAddHighLevelShaderMaterialFromFiles}$

Syntax

irr_mesh = IrrAddHighLevelShaderMaterialFromFiles (

vertex_program_filename as zstring ptr, vertex_start_function as zstring ptr,

vertex_prog_type as uinteger, pixel_program_filename as zstring ptr, pixel_start_function as zstring ptr, pixel_prog_type as uinteger, material_type as uinteger)

Description

Creates a new material using a high level shading language stored in files.

vertex program_filename: String containing the filename of the vertex shader program. This can be 0 if no vertex program shall be used.

vertex_start_function: Name of the entry function of the vertex shader program vertex_program_type: Vertex shader version used to compile the GPU program pixel_program_filename: String containing the filename of the pixel shader program. This can be 0 if no pixel shader shall be used.

pixel_start_function: Entry name of the function of the pixel shader program pixel_program_type: Pixel shader version used to compile the GPU program baseMaterial: Base material which renderstates will be used to shade the material.

Returns a type that contains a material_type number that can be used to shade nodes with this new material. If the shader could not be created it will return 0.

Example

```
shader = IrrAddHighLevelShaderMaterialFromFiles ( _
    ".\media\wood.vertex", "main", IRR_EVST_VS_1_1, _
    ".\media\wood.pixel", "main", IRR_EPST_PS_1_1, _
    IRR_EMT_SOLID )
```

IrrAddShaderMaterial

Syntax

irr_shader = IrrAddShaderMaterial (vertex_program as zstring ptr, pixel_program as zstring ptr, material_type as uinteger)

Description

Creates a new material using a shading language program stored in a string.

vertex program: String containing the source of the vertex shader program. This can be 0 if no vertex program shall be used. For DX8 programs, the will always input registers look like this: v0: position, v1: normal, v2: color, v3: texture cooridnates, v4: texture coordinates 2 if available. For DX9 programs, you can manually set the registers using the dcl_ statements.

pixel_program: String containing the source of the pixel shader program. This can be 0 if no pixel shader shall be used.

baseMaterial: Base material which renderstates will be used to shade the material.

Return: Returns a type that contains a material_type number that can be used to shade nodes with this new material. If the shader could not be created it will return 0

Example

```
shader = IrrAddShaderMaterial ( vertex_program, pixel_program,
IRR_EMT_SOLID )
```

IrrAddShaderMaterialFromFiles

Syntax

irr_shader = IrrAddShaderMaterialFromFiles (vertex_program_filename as zstring ptr, pixel_program_filename as zstring ptr, material_type as uinteger)

Description

Creates a new material using a shading language program stored in files.

vertex program: String containing the source of the vertex shader program. This can be 0 if no vertex program shall be used. For DX8 programs, the will always input registers look like this: v0: position, v1: normal, v2: color, v3: texture cooridnates, v4: texture coordinates 2 if available. For DX9 programs, you can manually set the registers using the dcl_ statements.

pixel_program: String containing the source of the pixel shader program. This can be 0 if no pixel shader shall be used.

baseMaterial: Base material which renderstates will be used to shade the material.

Return: Returns a type that contains a material_type number that can be used to shade nodes with this new material. If the shader could not be created it will return 0

```
shader = IrrAddShaderMaterialFromFiles ( ".\media\wood_low.vtx",
".\media\wood_low.pxl" IRR_EMT_SOLID )
```

IrrCreateNamedVertexShaderConstant

Syntax

result = IrrCreateNamedVertexShaderConstant (shader as IRR_SHADER, const_name as zstring ptr, const_preset as integer, const_data as single, data_count as integer)

Description

Creates a Vertex shader constant that allows you to change the value of a constant inside a shader during the execution of the program, simply assign one of the preset constants to the constant name or attach the constant to an array of floats and change the constant simply by changing the values in your array

Returns: 1 if the constant was sucessfully created

Example

```
IrrCreateNamedVertexShaderConstant ( shader, "Time", byval
IRR_NO_PRESET, @time, 1 )
```

IrrCreateNamedPixelShaderConstant

Syntax

result = IrrCreateNamedPixelShaderConstant (shader as IRR_SHADER, const_name as zstring ptr, const_preset as integer, const_data as single, data_count as integer)

Description

Creates a Pixel shader constant that allows you to change the value of a constant inside a shader during the execution of the program, simply assign one of the preset constants to the constant name or attach the constant to an array of floats and change the constant simply by changing the values in your array

Returns: 1 if the constant was sucessfully created

```
dim color(4) as Single => { 1.0, 1.0, 1.0, 1.0 }
IrrCreateNamedPixelShaderConstant ( shader, "color",
IRR_NO_PRESET, @color, 4 )
```

IrrCreateAddressedVertexShaderConstant

Syntax

result = IrrCreateAddressedVertexShaderConstant (shader as IRR_SHADER, const_address as integer, const_preset as integer, const_data as single, data_count as integer)

Description

Creates a Vertex shader constant that allows you to change the value of a constant inside a shader during the execution of the program, simply assign one of the preset constants to the constant name or attach the constant to an array of floats and change the constant simply by changing the values in your array

Returns: 1 if the constant was sucessfully created

Example

```
IrrCreateAddressedVertexShaderConstant ( shader, 4, IRR_NO_PRESET,
@time, 1 )
```

IrrCreateAddressedPixelShaderConstant

Syntax

result = IrrCreateAddressedPixelShaderConstant (shader as IRR_SHADER, const_address as integer, const_preset as integer, const_data as single, data_count as integer)

Description

Creates a Pixel shader constant that allows you to change the value of a constant inside a shader during the execution of the program, simply assign one of the preset constants to the constant name or attach the constant to an array of floats and change the constant simply by changing the values in your array

Returns: 1 if the constant was sucessfully created

```
dim position(3) as Single => { 0.0, 0.0, 0.0 }
IrrCreateAddressedPixelShaderConstant ( shader, 2, IRR_NO_PRESET,
@position, 3 )
```

IrrXEffectsStart

Syntax IrrXEffectsStart (vsm as integer = IRR_OFF, softShadows as integer = IRR_OFF, bitdepth32 as integer = IRR_OFF)
Description Starts the XEffects advanced shader extension provided by Bitplane from the Irrlicht Forums. This must be called before any other XEffects calls.
The first parameter 'vsm' is used to turn on the 'Variance Shadow Maps' feature. VSM is an advanced form of shading used to avoid aliasing problems that can be seen with the other shadowing function. It can create clear sharp shadowing. Use IRR_ON to enable this feature.
The second parameter 'soft shadows' provides blurred shadows, similar as those cast by a large source. Use IRR_ON to enable this feature.
The last parameter 'bit depth 32' enables 32 bit buffers for the internal processes. While this will use more video memory it can produce improved results.
Example
IrrXEffectsStart (IRR_OFF, IRR_ON)

IrrXEffectsEnableDepthPass

Syntax

IrrXEffectsEnableDepthPass(enable as integer)

Description

Enables a depth rendering pass. This is required for shaders that rely on depth information. Use IRR_ON to enable the function.

Example

IrrXEffectsEnableDepthPass (IRR_ON)

IrrXEffectsAddPostProcessingFromFile

Syntax

IrrXEffectsAddPostProcessingFromFile(name as zstring ptr, effectType as integer = 0)

Description

Adds a shader feature to the display from a GLSL or HLSL program stored in a file. Shaders do need some programming support so only the XEffects Shaders are supported through the XEffects calls.

The first parameter is the path and file name for the shader program. If you are operating in OpenGL you should use the GLSL extension and when operating in DirectX you shouhld use the HLSL extension.

The second parameter can usually be omitted or set to 0. Only when loading the SSAO shader (**not** the SSAO composite shader) should it be set to 1.

Example

IrrXEffectsAddPostProcessingFromFile (
"./media/shaders/ssao.glsl", 1)

IrrXEffectsSetPostProcessingUserTexture

Syntax

IrrXEffectsSetPostProcessingUserTexture(texture as irr_texture)

Description

Sets the user defined post processing texture. This is used internally for the SSAO shader but is used primarily for the water shader where it defines the specular surface pattern of the water.

You can change the texture through a sequence of images to produce an animated effect.

Example

IrrXEffectsSetPostProcessingUserTexture (waterTexture(i))

IrrXEffectsAddShadowToNode

Syntax

IrrXEffectsAddShadowToNode(node as irr_node, filterType as E_FILTER_TYPE = EFT_NONE, shadowType as E_SHADOW_MODE = ESM_BOTH)

Description

Adds the shadowing effect to a node. This controls both recieving and casting shadows.

The filterType defines the amount of sampling that is to be carried out on the node. This can be one of the following settings, increasing the filter increases the quality and also the cost of rendering.

EFT_NONE EFT_4PCF EFT_8PCF EFT_12PCF EFT_16PCF

The shadow type specifies the type of shadowing applied to the node. This can be set to one of the following settings: -

ESM_RECEIVE ESM_CAST ESM_BOTH ESM_EXCLUDE

Example

IrrXEffectsAddShadowToNode (roomNode)

IrrXEffectsRemoveShadowFromNode

Syntax

IrrXEffectsRemoveShadowFromNode(node as irr_node)

Description

Removes the shadowing effect from a node.

Example

IrrXEffectsRemoveShadowFromNode (roomNode)

IrrXEffects Exclude Node From Lighting Calculations

Syntax

IrrXEffectsExcludeNodeFromLightingCalculations(node as irr_node)

Description

Excludes a node from shadowing calculations.

Example

IrrXEffectsExcludeNodeFromLightingCalculations (particleNode)

IrrXEffectsAddNodeToDepthPass

Syntax

IrrXEffectsAddNodeToDepthPass(node as irr_node)

Description

Adds a node to the list of nodes used for calculating the depth pass.

Example

IrrXEffectsAddNodeToDepthPass (barrierNode)

IrrXEffectsSetAmbientColor

Syntax

IrrXEffectsSetAmbientColor(R as uinteger, G as uinteger, B as uinteger, Alpha as uinteger)

Description

Sets the ambient lighting procuded in the scene by the XEffects system.

Example

IrrXEffectsSetAmbientColor (32,32,32,0)

IrrXEffectsSetClearColor

Syntax

IrrXEffectsSetClearColor(R as uinteger, G as uinteger, B as uinteger, Alpha as uinteger)

Description

The XEffects system uses a different background color to the one specified in the IrrBeginScene call use this call to set this default background color.

Example

IrrXEffectsSetClearColor (255,250,32,0

IrrXEffectsAddShadowLight

Syntax

IrrXEffectsAddShadowLight(

shadowDimen as uinteger,

posX as single, byVal posY as single, byVal posZ as single, targetX as single, byVal targetY as single, byVal targetZ as single, R as single, byval G as single, byval B as single, byval Alpha as single, lightNearDist as single, byval lightFarDist as single, angleDegrees as single)

Description

Adds a special dynamic shadow casting light to the scene, for each of these lights that you add there is a seperate shadow map created and a seperate render pass so for each light you add the scene takes more memory and gets slower.

The first parameter specifies the shadow map resolution for the shadow light. The shadow map is always square, so you need only pass 1 dimension, preferably a power of two between 512 and 2048, maybe larger depending on your quality requirements and target hardware.

The pos parameters specify the lights initial position

The target parameters is the (look at) target for the light

The color setting are the floating point color intensity values of the light The near and far distance of the light are very important values for determining the reach of the light. The last parameter is the FOV (Field of view), since the light is similar to a spot light, the field of view will determine its area of influence. Anything that is outside of a lights frustum (Too close, too far, or outside of it's field of view) will be unlit by this particular light, similar to how a spot light works.

Example

```
IrrXEffectsAddShadowLight( 512, 200,200,0, 0,0,0, _
0.7,0.7,0.6,0.0, 1.0, 1200.0, 89.99 )
```

IrrXEffectsSetShadowLightPosition

Syntax

IrrXEffectsSetShadowLightPosition(lightIndex as uinteger, posX as single, byVal posY as single, byVal posZ as single)

Description

Set the position of a shadow light. the index refers to the numerical order in which the lights were added.

Example

IrrXEffectsSetShadowLightPosition (0, 200,200,0)

IrrXEffectsGetShadowLightPosition

Syntax

IrrXEffectsGetShadowLightPosition(lightIndex as uinteger, posX as single, byVal posY as single, byVal posZ as single)

Description

Get the position of a shadow light. the index refers to the numerical order in which the lights were added.

Example

IrrXEffectsGetShadowLightPosition (0, x, y, z)

IrrXEffectsSetShadowLightTarget

Syntax

IrrXEffectsSetShadowLightTarget(lightIndex as uinteger, targetX as single, byVal targetY as single, byVal targetZ as single)

Description

Set the target location of a shadow light. the index refers to the numerical order in which the lights were added.

Example

IrrXEffectsSetShadowLightTarget (0, 25,15,0)

IrrXEffectsGetShadowLightTarget

Syntax

IrrXEffectsGetShadowLightTarget(lightIndex as uinteger, targetX as single, byVal targetY as single, byVal targetZ as single)

Description

Get the target location of a shadow light. the index refers to the numerical order in which the lights were added.

Example

IrrXEffectsGetShadowLightTarget (0, x, y, z)

IrrXEffectsSetShadowLightColor

Syntax

IrrXEffectsSetShadowLightColor(lightIndex as uinteger,

R as single, byval G as single, byval B as single, byval Alpha as single)

Description

Set the target location of a shadow light. the index refers to the numerical order in which the lights were added.

Example

IrrXEffectsSetShadowLightColor (0, 1.0, 0.75, 0.2, 0.0)

IrrXEffectsGetShadowLightColor

Syntax IrrXEffectsGetShadowLightColor(lightIndex as uinteger, R as single, byval G as single, byval B as single, byval Alpha as single)

Description

Get the target location of a shadow light. the index refers to the numerical order in which the lights were added.

Example

IrrXEffectsGetShadowLightColor (0, r, g, b, a)

Scene

Calls for managing the scene, loading and creating mesh objects and then adding them to the scene as nodes to be rendered on the screen.

IrrGetRootSceneNode

Syntax irr_node = IrrGetRootSceneNode()

Description Gets the scenes root node, **all** scene nodes are children of this node

Example

TheScene = IrrGetRootSceneNode()

IrrGetMesh

Syntax

irr_mesh = IrrGetMesh(Filename of the mesh object to load as zstring)

Description

Loads the specified mesh ready to be added to the scene. The Irrlicht engine supports a wide range of mesh types including BSP, MD2, 3DS, Direct X, etc...

Example

DolphinMesh = IrrGetMesh("Dolphin.x")

IrrCreateMesh

Syntax

IrrCreateMesh (mesh_name as zstring ptr, vertex_count as integer, vertices as IRR_VERT, indices_count as integer, indices as ushort) as irr_mesh

Description

Create a new mesh. You must supply a list of vertices of type IRR_VECT and an array of indices that refer to these vertices. The indices are taken in groups of three joining up the dots defined by the veticies and forming a collection of triangles.

Example

```
PyramidMesh = IrrCreateMesh( "Pyramid", 5, vertices(0), 18,
indicies(0))
```

IrrAddHillPlaneMesh

Syntax

IrrAddHillPlaneMesh (mesh_name As zString Ptr, tileSizeX As Single, tileSizeY As Single, tileCountX As Integer, tileCountY As Integer, material As uInteger Ptr = 0, hillHeight As Single = 0, countHillsX As Single = 0, countHillsY As Single = 0, textureRepeatCountX As Single = 1, textureRepeatCountY As Single = 1) as irr_mesh

Description

Creates a hill plane mesh that represents a simple terrain. Many properties have default values allowing a mesh to be created with a simple call

Example

TerrainMesh = IrrAddHillPlaneMesh("Terrain", 1.0, 1.0, 10, 10)

IrrWriteMesh

Syntax

IrrWriteMesh(mesh as irr_mesh, file_format as IRR_MESH_FILE_FORMAT, save_filename as zstring) as uinteger

Description

Write the first frame of the supplied animated mesh out to a file using the specified file format. The following file formats are supported by Irrlicht: -

Irrlicht Native mesh writer, for static .irrmesh files. EMWT_IRR_MESH COLLADA mesh writer for .dae and .xml files. EMWT_COLLADA

STL mesh writer for .stl files. EMWT_STL

The function will return the follow error codes: -

(0) Could not get mesh writer object

(1) Could not open file

(2) Unable to write the mesh to the file

(3) Successfully wrote file

Example

IrrRemoveMesh

Syntax

IrrRemoveMesh(mesh as irr_mesh)

Description

Removes a mesh from the scene cache, freeing up resources.

Example

IrrRemoveMesh(my_mesh)

IrrRenameMesh

Syntax

IrrRenameMesh(mesh as irr_mesh, name as zstring ptr)

Description Rename a loaded mesh through the scene cache, the mesh can then subsequently be loaded again as a different mesh Example

IrrRenameMesh(my_mesh, "New Name"

IrrClearUnusedMeshes

Syntax IrrClearUnusedMeshes()

Description

Clears all meshes that are held in the mesh cache but not used anywhere else. Any references to these meshes will become invalid.

Example

IrrClearUnusedMeshes()

IrrSetMeshHardwareAccelerated

Syntax

IrrSetMeshHardwareAccelerated (mesh as irr_mesh, frame number as integer)

Description

Set the supplied mesh as a Hardware Accelerated object, this offloads the verticies and indicies to hardware support on the graphics card, making the process of rendering those meshes much faster. The feature must be supported on the graphics card and the object must contain over 500 vertices for the operation to be successful. This operation is applied to all mesh buffers in the mesh.

Example

IrrSetMeshHardwareAccelerated(ShipMesh, 0)

IrrCreateBatchingMesh

Syntax IrrCreateBatchingMesh ()

Description Create a batching mesh that will be a collection of other meshes into a single source mesh. The function of the batching mesh is to avoid the use of large numbers of nodes that adds an overhead to the rendering process that can significantly slow it down. Where you have a forest with a thousand trees you will see a significant increase in performance by batching all of those trees into a smaller number of node.

Returns: A bactching mesh, while this is handled as an irr_mesh it should only be used with batching mesh commands.

Example

```
batchingMesh = IrrCreateBatchingMesh( )
```

IrrAddToBatchingMesh

Syntax

IrrAddToBatchingMesh (

meshBatch as irr_mesh,

mesh as irr_mesh,

posX as single = 0.0f, posY as single = 0.0f, posZ as single = 0.0f,

rotX as single = 0.0f, rotY as single = 0.0f, rotZ as single = 0.0f,

scaleX as single = 1.0f, scaleY as single = 1.0f, scaleZ as single = 1.0f)

Description

Adds a mesh to the batching mesh at the specified position, rotation and scale. If each of your meshes requires a different texture you should call IrrSetMeshMaterialTexture for the mesh you are about to add prior to adding the mesh to the batch.

Example

IrrAddToBatchingMesh(batchingMesh, treeMesh)

IrrFinalizeBatchingMesh

Syntax

IrrFinalizeBatchingMesh (mesh as irr_mesh, frame number as integer)

Description

Finalises the batching mesh, this should be called once all of the meshes have

been added to the batching mesh. The function returns a new mesh object that can be used in all standard mesh calls..

Example

Dim as irr_mesh newMesh = IrrFinalizeBatchingMesh(BatchingMesh)

IrrGetMeshFrameCount

Syntax

integer = IrrGetMeshFrameCount (mesh as irr_mesh)

Description

Gets the number of frames in the supplied mesh. You can use this value to traverse the indicies and vertices in a mesh containing a number of frames.

Example

MeshFrameCount = IrrGetMeshFrameCount(WolfMesh)

IrrGetMeshBufferCount

Syntax

integer = IrrGetMeshBufferCount (mesh as irr_mesh, frame number as integer
)

Description

Gets the number of mesh buffers in the supplied mesh. You can use this value to traverse the indicies and vertices in a mesh containing a number of mesh buffers. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0.

Most meshes only contain one mesh buffer however the artist creating the mesh may decide to break the mesh up into a number of groups of meshes, for example a house might have a roof mesh buffer and a walls mesh buffer.

Example

MeshBufferCount = IrrGetMeshBufferCount(TankMesh, 0)

IrrGetMeshIndexCount

Syntax

integer = IrrGetMeshIndexCount (mesh as irr_mesh, frame number as integer, mesh_buffer as integer)

Description

Gets the number of Indicies in the supplied mesh. You can use this value to allocate an array for reading out the list of indicies in a mesh. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0. If the mesh contains a number of mesh buffers you can specific which mesh buffer you want to access, if you omit this parameter mesh buffer 0 will be used.

Example

MeshIndexCount = IrrGetMeshIndexCount(MapMesh, 0,0)

IrrGetMeshIndices

Syntax

IrrGetMeshIndices (mesh as irr_mesh, frame number as integer , indicies as ushort, mesh_buffer as integer)

Description

Gets the list of indices in a mesh and copies them into the supplied buffer. Each index references a vertex in the mesh the indices are grouped into three's and together form a triangular surface. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0. If the mesh contains a number of mesh buffers you can specific which mesh buffer you want to access, if you omit this parameter mesh buffer 0 will be used.

Indices should be the first element of an array or the first integer in a pool of allocated memory, it is passed into the wrapper by reference as a pointer. You must ensure that the array you supply is large enough to contain all of the indices otherwise an overflow will occur and memory will be corrupted.

Example

IrrGetMeshIndices(MapMesh, 0, Indicies(0),0)

IrrSetMeshIndices

Syntax

IrrSetMeshIndices(mesh as irr_mesh, frame number as integer , indicies as ushort, mesh_buffer as integer)

Description

This sets the value of the list of indicies in a mesh copying them from the supplied buffer. Each index references a vertex in the mesh the indices are grouped into three's and together form a triangular surface. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0. If the mesh contains a number of mesh buffers you can specific which mesh buffer you want to access, if you omit this parameter mesh buffer 0 will be used.

Indices should be the first element of an array or the first integer in a pool of allocated memory, it is passed into the wrapper by reference as a pointer. You must ensure that the array you supply is large enough to contain all of the indices otherwise an overflow will occur and erroneous values will be written into the mesh causing unpredictable results.

Example

IrrSetMeshIndices(MapMesh, 0, Indicies(0),0)

IrrGetMeshVertexCount

Syntax

integer = IrrGetMeshVertexCount (mesh as irr_mesh, frame number as integer, mesh_buffer as integer)

Description

Gets the number of Vertices in the supplied mesh. You can use this value to allocate an array for reading out the list of vertices in a mesh. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0. If the mesh contains a number of mesh buffers you can specific which mesh buffer you want to access, if you omit this parameter mesh buffer 0 will be used.

Example

MeshVertexCount = IrrGetMeshVertexCount(MapMesh, 0)

IrrGetMeshVertexMemory

Syntax

IrrGetMeshVertexMemory (mesh as irr_mesh, frame number as integer , mesh_buffer as integer)

Description

Get a memory pointer to the vertex memory for the supplied mesh operations can be carried out very quickly on vertices through this function but object sizes and array access needs to be handled by the caller.

Example

Dim as IRR_VERT verts = IrrGetMeshVertexMemory(MapMesh, 0, 0)

IrrGetMeshVertices

Syntax

IrrGetMeshVertices (mesh as irr_mesh, frame number as integer , vertices as IRR_VERT, mesh_buffer as integer)

Description

Gets the list of vertices in a mesh and copies them into the supplied buffer. Each vertex represents a point in the mesh that is the corner of one of the grou pof triangles that is used to construct the mesh. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0. If the mesh contains a number of mesh buffers you can specific which mesh buffer you want to access, if you omit this parameter mesh buffer 0 will be used.

Vertices should be the first element of an array or the first IRR_VERT structure in a pool of allocated memory, it is passed into the wrapper by reference as a pointer. You must ensure that the array you supply is large enough to contain all of the vertices otherwise an overflow will occur and memory will be corrupted.

Example

IrrSetMeshVertices

Syntax

IrrSetMeshVertices(mesh as irr_mesh, frame number as integer , indicies as IRR_VERT, mesh_buffer as integer)

Description

This sets the value of the list of vertices in a mesh copying them from the supplied buffer. Each vertex represents a point in the mesh that is the corner of one of the grou pof triangles that is used to construct the mesh. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0. If the mesh contains a number of mesh buffers you can specific which mesh buffer you want to access, if you omit this parameter mesh buffer 0 will be used.

Vertices should be the first element of an array or the first IRR_VERT structure in a pool of allocated memory, it is passed into the wrapper by reference as a pointer. You must ensure that the array you supply is large enough to contain all of the vertices otherwise an overflow will occur and erroneous values will be written into the mesh causing unpredictable results.

Example

IrrSetMeshVertices(MapMesh, 0, Vertices(0), 0)

IrrSetNodeMesh

Syntax

IrrSetNodeMesh (node as irr_node, mesh as irr_mesh)

Description

Sets the mesh used by a node creaed from a mesh model.

Example

IrrSetNodeMesh(BuildingNode, LowDetailMesh)

IrrScaleMesh

Syntax

IrrScaleMesh(mesh as irr_mesh, scale as single, frame number as integer = 0, mesh_buffer as integer = 0, source mesh as irr_mesh = 0)

Description

Scales the verticies in a mesh without affecting the normals, tangents or texture co-ordinates. This is particularly useful for enlarging a mesh without affecting lighting. It should be noted though that scaling the mesh will scale all of the nodes that use it as their source. The scaling is applied unformly to all axis.

Example

IrrScaleMesh(StatueMesh, 2.0)

IrrSetMeshMaterialTexture

Syntax

IrrSetMeshMaterialTexture(

mesh as irr_mesh,

byval texture as irr_texture,

byval material_index as integer,

byval buffer as integer = 0)

Description

Apply the supplied texture the specified mesh. Upto four textures can be applied to the material by applying them to different material indicies, these textures can be used by materials or shader functions. Setting a mesh texture will apply the texture to all nodes that use that mesh it can also used for texturing a mesh before it is added to a batch mesh.

Example

IrrSetMeshMaterialTexture(StatueMesh, stoneTexture, 0)

IrrSetMeshVertexColors

Syntax

IrrSetMeshVertexColors(mesh as irr_mesh, frame number as integer , vertexColour as uinteger ptr, vertexGroupStartIndicies as uinteger ptr,

vertexGroupEndIndicies as uinteger ptr, numberOfGroups as uinteger, mesh_buffer as integer)

Description

This sets the color of groups of verticies in a mesh. You can define any number of groups of verticies and set the color of those group invividually. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0. If the mesh contains a number of mesh buffers you can specific which mesh buffer you want to access, if you omit this parameter mesh buffer 0 will be used.

```
Example
```

```
DIM color(0 to 2) as uinteger
color(0) = RGBA(255,0,0,0)
color(1) = RGBA(255,0,0,0)
color(2) = RGBA(255,0,0,0)
DIM start as uinteger = 0
DIM end as uinteger = 2
IrrSetMeshVertexColors( MapMesh, 0, @color, @start, @end, 1, 0 )
```

IrrSetMeshVertexCoords

Syntax

IrrSetMeshVertexCoords(mesh as irr_mesh, frame number as integer , vertexCoordinates as IRR_VECTOR Ptr, vertexGroupStartIndicies as uinteger ptr, vertexGroupEndIndicies as uinteger ptr, numberOfGroups as uinteger, mesh_buffer as integer)

Description

This sets the co-ordinates of groups of verticies in a mesh. You can define any number of groups of verticies and set the color of those group invividually. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0. If the mesh contains a number of mesh buffers you can specific which mesh buffer you want to access, if you omit this parameter mesh buffer 0 will be used.

Example

```
DIM pos(0 to 2) as IRR_VECTOR
pos(0).x = 0 : pos(0).y = 0 : pos(0).z = 0
pos(1).x = 1 : pos(1).y = 0 : pos(1).z = 0
```

```
pos(2).x = 0 : pos(2).y = 1 : pos(2).z = 0
DIM start as uinteger = 0
DIM end as uinteger = 2
IrrSetMeshVertexCoords( MapMesh, 0, @color, @start, @end, 1, 0 )
```

IrrSetMeshVertexSingleColor

Syntax

IrrSetMeshVertexSingleColor(mesh as irr_mesh, frame number as integer, vertexColour as uinteger ptr, vertexGroupStartIndicies as uinteger ptr, vertexGroupEndIndicies as uinteger ptr, numberOfGroups as uinteger, mesh_buffer as integer)

Description

This sets the color of groups of verticies in a mesh. You can define any number of groups of verticies and set the color of those group invividually. If the mesh is animated frame number indicates the number of the frame to recover mesh data for if it is not animated this value should be set to 0. If the mesh contains a number of mesh buffers you can specific which mesh buffer you want to access, if you omit this parameter mesh buffer 0 will be used.

Example

```
DIM start as uinteger = 0
DIM end as uinteger = 2
IrrSetMeshVertexSingleColor( MapMesh, 0, RGBA(255,255,255,255),
@start, @end, 1, 0 )
```

IrrGetMeshBoundingBox

Syntax

IrrGetMeshBoundingBox(mesh as irr_mesh, min X as single, min Y as single, min Z as single, min X as single, min Y as single, min Z as single)

Description

Gets the bounding box of a mesh into the supplied variables, the six paramters define the corners of an axis aligned cube that contains the whole mesh.

Example

```
IrrGetMeshBoundingBox( MapMesh, topX, topY, topZ, bottomX,
bottomY, bottomZ )
```

IrrAddMeshToScene

Syntax

irr_node = IrrAddMeshToScene(Mesh object as irr_mesh)

Description Adds a mesh to the scene as a new 3D 'node'.

Example

DolphinMesh = IrrGetMesh("Dolphin.x") SceneNode = IrrAddMeshToScene(DolphinMesh)

IrrAddMeshToSceneAsOcttree

Syntax

irr_node = IrrAddMeshToSceneAsOcttree (Mesh object as irr_mesh)

Description

Adds a mesh to the scene as a new 3D 'node'. This method optimise's the mesh with an Octtree, this is particularly useful for maps where there is a lot of geometry in the mesh but little of it can be seen at any one time. Optimizing your node with this function will result in a large increase in performance.

Example

```
MapMesh = IrrGetMesh( "ctfblue.bsp" )
MapNode = IrrAddMeshToSceneAsOcttree( MapMesh )
```

IrrAddStaticMeshForNormalMappingToScene

Syntax

irr_node = IrrAddStaticMeshForNormalMappingToScene(Mesh object as irr_mesh)

Description

Adds a mesh to the scene as a static object, the mesh is altered so that it is suitable for the application of a Normal or Parallax mapping material, any animation information is lost.

Example

```
StoneRoomMesh = IrrGetMesh( "StoneRoom.x" )
SceneNode = IrrAddStaticMeshForNormalMappingToScene( StoneRoomMesh
)
IrrSetNodeMaterialTexture( SceneNode, colorMap, 0 )
IrrSetNodeMaterialTexture( SceneNode, normalMap, 1 )
IrrMaterialSetSpecularColor( IrrGetMaterial( SceneNode ), 0, 0, 0
)
IrrSetNodeMaterialType( SceneNode, EMT_PARALLAX_MAP_SOLID )
' adjust the height of the paralax effect
IrrMaterialSetMaterialTypeParam( IrrGetMaterial( SceneNode ), 0, 0, 0.035f )
```

IrrLoadScene

Syntax

IrrLoadScene (file_name As zString Ptr)

Description

Loads all meshes and creates nodes for a scene defined within a file created by IrrEdit.

Example

IrrLoadScene("Map1.irr")

IrrSaveScene

Syntax IrrSaveScene (file_name As zString Ptr)

Description Saves the current scene into a file that can be loaded by irrEdit.

Example

IrrSaveScene("MyScene.irr")

IrrGetSceneNodeFromId

Syntax

Irr_node = IrrGetSceneNodeFromId(id as integer)

Description

Get a scene node based on its ID and returns null if no node is found. This is particularly useful for obtaining references to nodes created automatically when using IrrLoadScene.

Example

IrrNode = IrrGetSceneNodeFromId(15)

IrrGetSceneNodeFromName

Syntax

Irr_node = IrrGetSceneNodeFromId(id as zstring ptr)

Description

Get a scene node based on its name and returns null if no node is found. This is particularly useful for obtaining references to nodes created automatically when using IrrLoadScene.

Example

IrrNode = IrrGetSceneNodeFromName("Box")

IrrAddBillBoardToScene

Syntax

irr_node = IrrAddBillBoardToScene (X size of the node as integer, Y size of the node as integer, X position as integer, Y position as integer, Z position as integer)

Description

Adds a billboard to the scene of the specified size and at the specified position. A billboard is a flat 3D textured sprite that always faces towards the camera. You need to texture this element with a separate command.

Example

Billboard = IrrAddBillBoardToScene(10.0,8.0, 0,0,0)

IrrSetBillBoardColor

Syntax

irr_node = IrrSetBillBoardColor (node as irr_node, topColor as uinteger, bottomColor as integer)

Description

Set the color of the top and bottom verticies in a billboard applying a vertical graduated shade to it. The colors should be generated with the FreeBasic RGBA function

Example

```
IrrSetBillBoardColor( Billboard, RGBA(255,255,255,255),
RGBA(0,0,0,0))
```

IrrSetBillBoardSize

Syntax

irr_node = IrrSetBillBoardSize (node as irr_node, BillWidth as single, BillHeight as single)

Description

Adds a billboard to the scene of the specified size and at the specified position. A billboard is a flat 3D textured sprite that always faces towards the camera. You need to texture this element with a separate command.

Example

IrrSetBillBoardSize(Billboard, 10.0, 8.0)

IrrAddBillboardTextSceneNode

Syntax

irr_node = IrrAddBillboardTextSceneNode (font as irr_font, text as wstring, X size of the node as integer, Y size of the node as integer, X position as integer, Y position as integer, Z position as integer, parent as irr_node, topColor as uinteger, bottomColor as uinteger)

Description Adds a text billboard to the scene of the specified size and at the specified

position. A text billboard is a flat 3D textured sprite that always faces towards the camera and has the supplied text written onto it. You should not texture this element.
font defines the font that is used to generate the text.
text is a wide character string containing the text you want to display on the billboard.
X_size and Y_size define the width and height of the billboard
X, Y and Z define the position of the billboard.
Parent defines the object that is the parent to this billboard, if there is no parent this should be set to IRR_NO_OBJECT
topColor is the colour value of the top of the text on the billboard. This can be created with hte RGBA command.
bottomColor is the colour value of the bottom of the text on the billboard. This can be created with hte RGBA command.
Example
Billboard = IrrAddBillboardTextSceneNode(_ font, "Hello World", _ 64.0, 12.0, _ 0.0, 40.0, 0.0, _ parentNode, _

IrrAddParticleSystemToScene

RGBA(255,255,0,0), _ RGBA(255,0,0,255))

Syntax

node as irr_particle_system =IrrAddParticleSystemToScene (add_emitter)

Description

Adds a particle system to the scene as a node, a particle system is an object that creates and manages hundreds of small billboard like objects that are used to represent smoke, rain and other natural effects. Once created you then need to

add emitters and affecters to create and control the particles.

Add emitter can be one of the following values: -IRR_NO_EMITTER For no default emitter (this is probably the option you will use and you will then add a specific emitter later) IRR_DEFAULT_EMITTER To create a default emitter that ejects a thin vertical stream of particles.

Example

Smoke = IrrAddParticleSystemToScene(IRR_NO_EMITTER)

IrrAddSkyBoxToScene

Syntax

irr_node = IrrSkyBoxToScene (up_texture as irr_texture, down_texture as irr_texture, left_texture as irr_texture, right_texture as irr_texture, front_texture as irr_texture, back_texture as irr_texture)

Description

Adds a skybox node to the scene this is huge hollow cube that encapsulates the entire scene and has a different texture applied to each of its six surfaces to represent a distant sky or matte scene.

Example

```
SkyBox = IrrAddSkyBoxToScene( _
    IrrGetTexture("./media/irrlicht2_up.jpg"),_
    IrrGetTexture("./media/irrlicht2_dn.jpg"),_
    IrrGetTexture("./media/irrlicht2_lf.jpg"),_
    IrrGetTexture("./media/irrlicht2_rt.jpg"),_
    IrrGetTexture("./media/irrlicht2_ft.jpg"),_
    IrrGetTexture("./media/irrlicht2_bk.jpg"))
```

IrrAddSkyDomeToScene (contributed by Eponasoft)

Syntax

irr_node = IrrAddSkyDomeToScene (texture as irr_texture, horizontal_res as uinteger, vertical_res as uinteger, texture_percentage as double, sphere_percentage as double, sphere_radius as single)

Description

Adds a skydome node to the scene this is huge hollow sphere (or part of a sphere) that encapsulates the entire scene to represent a distant sky or matte scene. The horizontal and vertical resolution define the number of segments in the mesh of the sphere (setting these too high can quickly produce a very costly mesh). Texture percentage defines the amount of the texture that is mapped to the scene, this should be a value between 0 and 1 (0 being non of the texture and 1 being the whole texture). Finally sphere percentage defines how much of a sphere is created and should be a value between 0 and 2 (0 being none of a sphere, 1 being a hemi-sphere and 2 being a full sphere).

Example

SkyBox = IrrAddSkyDomeToScene(IrrGetTexture("./media/domesky.jpg"), 8, 8, 1.0, 2.0, 10000.0);

IrrAddEmptySceneNode

Syntax

irr_node = IrrAddEmptySceneNode

Description

Adds an empty node to the scene. This is required if you wish to add custom OpenGL commands with no Irrlicht Objects.

Example

EmptyNode = IrrAddEmptySceneNode

IrrAddTestSceneNode

Syntax

irr_node = IrrAddTestSceneNode

Description

Adds a simple cube object to the scene. This is particularly useful for testing and is a quick and easy way of playing objects into the scene for testing placement.

Example

IrrAddCubeSceneNode

Syntax

irr_node = IrrAddCubeSceneNode(size as single)

Description

Adds a simple cube object to the scene with the specified dimensions.

Example

MyCube = IrrAddCubeSceneNode(10.0)

IrrAddSphereSceneNode

Syntax

irr_node = IrrAddSphereSceneNode(radius as single, poly_count as integer)

Description

Adds a simple sphere object to the scene of the specified radius and with the specified level of detail. A reasonable value for poly_count would be 16 setting this value too high could produce a very high density mesh and affect your frame rate adversely.

Example

MySphere = IrrAddSphereSceneNode(0.5, 16)

IrrAddWaterSurfaceSceneNode (contributed by Eponasoft)

Syntax

irr_node = IrrAddWaterSurfaceSceneNode (mesh as irr_mesh, waveHeight as Single = 2.0, waveSpeed as Single = 300.0, waveLength as Single = 10.0, parent as irr_scene_node = 0, id as Integer = -1, positionX as Single = 0, positionY as Single = 0, positionZ as Single = 0, rotationX as Single = 0, rotationY as Single = 0, rotationZ as Single = 0, scaleX as Single = 1.0, scaleY as Single = 1.0, scaleZ as Single = 1.0)

Description

Adds a mesh with a water animator applied to it, the mesh is animated automatically to simulate a water effect across its surface. Many properties are predefined for this node and a convincing water effect can be created simply by supplying the parameter for the mesh, however the node can be positioned, rotated and scaled by this call and the appearance of the waves on its surface can be adjusted.

Example

IrrAddZoneManager

Syntax

irr_node = IrrAddZoneManager (initialNearDistance as single, initialFarDistance as single)

Description

Adds a zone/distance management node to the scene. This simple but very powerful object has no visible geometry in the scene, it is used by attaching other nodes to it as children. When the camera is further away than the far distance and closer than the near distance to the zone manager all of the zones child objects are made invisible. This allows you to group objects together and automatically have them hidden from the scene when they are too far away to see. By using the near distance you could have two sets of objects in the scene one with high detail for when you are close and another with low detail for when you are far away.

Another way to use the zone manager would be to test when your camera is inside the zones bounding box and switch its visibility on and off manually.

Example

zone = IrrAddZoneManager(100,300)

IrrAddClouds (Node by G Davidson)

Syntax

irr_node = IrrAddClouds (texture as irr_texture, lod as uinteger, depth as uinteger, density as uinteger) Description

Adds a set of clouds to the scene. These clouds are most appropriate to a cloud effect experienced by a vehicle flying through them and could be of particular use in masking the transition of a spacecraft from an orbital vantage point to a flat terrain object. They do make a nice ordernary cloud effect too but can appear unrealistic when they are directly over the observer.

LOD defeines the level of detail in the cloud, more detail is added into the cloud depending on the distance of the observer from the object. depth defines the depth of recursion when creating the cloud and finally density defines the number of clouds in the cloud object.

Example

CloudNode = IrrAddClouds(CloudTexture, 3, 1, 500)

IrrAddLensFlare (Node by Paulo Oliveira with updates from gammaray and torleif)

Syntax

irr_node = IrrAddLensFlare (texture as irr_texture)

Description

Adds a lens flare patch object to the scene, this object simulates the effect of bright lights on the optics of a camera., the position of the lens flare can be set and changed with the IrrSetNodePosition command. The lens flare obejct uses a bitmap containing a series of 128x128 images representing stages of the the lens flare effect.

Example

SceneNode = IrrAddLensFlare(LensTexture)

IrrAddGrass (Node by G Davidson)

Syntax

irr_node = IrrAddGrass (terrain as irr_terrain, x as integer, y as integer, patchSize as integer, fadeDistance as single, crossed as integer, grassScale as single, maxDensity as uinteger, dataPositionX as integer, dataPositionY as integer, heightMap as irr_image, textureMap as irr_image, grassMap as irr_image, grassTexture as irr_texture)

Description

Adds a grass object to the scene. Grass objects are associated with terrain and tile terrain objects and are used to place small billboard objects into the scene representing folliage, this implementation of grass creates a large number of grass objects already positioned across the terrain and then dynamically shows or hides them depending on where the camera is within the scene. The grass is also affected with a wind modifier that gently moves the grass as if it were caught in the wind (by setting the speed of the wind to zero the grass will become static and you will see an increase in performance).

The position and size of the patch of grass can be set with x, y, patchSize and grassScale.

FadeDistance controls the distance at which the number of displayed grass elements in that patch are reduced. If this is set to 1.0 then when the cameral is inside the patch all of grass will be displayed but once outside less and less will be shown. By increasing this to 2.0 then all of the grass is shown until the camera is two patches distant. This gives a better appearence but reduces performance as more grass has to be drawn.

crossed can be set to either IRR_ON or IRR_OFF. When off each piece of grass is a separate entity with its own position and rotation. When On grass is paired up and placed to form a cross. Crossed grass can have a better appearance as you rotate around it. However individual grass can give the impression that there is more of it and you can therefore reduce the number of grass blades and increase performance.

MaxDensity controls the number of individual clumps of folliage that are created.

Dataposition X and Y can be used with a large bitmap associated with a tiled terrain and allow the color information to be taken from an offset position on the bitmap.

Heightmap is an image that contains the height of the terrain onto which the grass is placed.

TextureMap is the color map used to color the verticies of the grass and allow you to create areas of dark of light grass, you can use the terrain color map here.

GrassMap is an image used to adjust the height and density of the grass. For example you might have a patch where you dont want to see any grass or a

barren patch where you want short stubble.

GrassTexture is the actually texture used for the grass. This RGBA image is automatically broken up into a number of sections that are used to texure different clumps of grass.

Grass usually looks best when it is closely matched to the color of the terrain and to assist with this a new Material Type has been added IRR_EMT_TRANSPARENT_ADD_ALPHA_CHANNEL_REF that adds the color of grass texture to the color of the grass which is automatically set to the color of the terrain that it lies upon.

Example

```
grassNode = IrrAddGrass ( Terrain, x, y, 1024, 1.0, 250, 0, 0,
terrainHeight, terrainColor, grassMap, grassTexture )
```

IrrSetShadowColor

Syntax

IrrSetShadowColor (Alpha as integer, Red as integer, Green as integer, Blue as integer)

Description

Sets the color of shadows cast by objects in the scene. If you are observing a bright scene you might use a light grey shadow instead of a heavy black shadow to add to realism.

Example

IrrSetShadowColor(0, 128, 128, 128)

IrrSetFog

Syntax

irr_node = IrrSetFog (Red as integer, Green as integer, Blue as integer, fogtype
as integer, fog_start as single, fog_end as single, density as single)

Description Set the properties of fog in the scene. Red, Green and Blue define the fog color, you should set this to the same color as your sky so the scene fogs out nicely into nothing. These are integer values in the range of 0 to 255

Fogtype specifies whether you want the fog to increase in a linear mannar or exponentially - exponential fog usually

looks more atmospheric while linear looks more like a dense sea fog. This may be specified as either IRR LINEAR FOG

IRR_EXPONENTIAL_FOG

Fog start and end specify the distance at which the fog starts and the distance at which the fog reaches its maximum density. The values here will depend on the size and scale of the scene.

Density is only used with exponential fog and determines how quickly the exponential change takes place, good values for this range from 0 to 1

Example

ThinFog = IrrSetFog (240,255,255, IRR_EXPONENTIAL_FOG, 0.0,8000.0, 0.5)

IrrDraw3DLine

Syntax

IrrDraw3DLine(x_start as single, y_start as single, z_start as single, x_end as single, y_end as single, z_end as single, Red as integer, Green as integer, Blue as integer)

Description

Draws a line onto the display using 3D co-ordinates and a specified color.

Example

```
IrrBeginScene( 240, 255, 255 )
IrrDraw3DLine( 0.0, 0.0, 0.0, 0.0, 50.0, 0.0, 0, 255, 0 )
IrrDrawScene
```

IrrSetSkyDomeColor

Syntax

IrrSetSkyDomeColor(dome as irr_node, horizontalRed as uinteger, horizontalGreen as uinteger, horizontalBlue as uinteger, zenithRed as uinteger, zenithGreen as uinteger, zenithBlue as uinteger)

Description

Set the color of the verticies in the skydome. Two colors are defined one for the horizon and another for the top of the sky dome, this simulates the type of coloring effects you see in the sky. If you are using a full spherical skydome the horizontal color will be the color at the bottom of the skydome.

Example

' color the skydome so that it is brighter at the horizon and a darker blue at the top of the sky IrrSetSkyDomeColor(SkyDome, 128, 128, 255, 64, 64, 255)

IrrSetSkyDomeColorBand

Syntax

IrrSetSkyDomeColorBand(dome as irr_node, horizontalRed as uinteger, horizontalGreen as uinteger, horizontalBlue as uinteger, bandVerticalPosition as integer, bandFade as single, addative as uinteger)

Description

Creates a horizontal band of color in the skydome, this is mainly useful for creating additional bands of color at the horizon, where your sky is a graduation of blues and then in the morning you have a brighter golden band as the sun rises. The vertical position in the vertex at which you wish to create the band, bandFade defines the amount that the band is faded into the existing skydome color, addative can be IRR_ON to add the color of the band to the existing color of the skydome or IRR_OFF to replace it.

Example

' add a band of golden color at the horizon IrrSetSkyDomeColorBand (SkyDome, 240,220,128, 24, 0.25, IRR_ON)

IrrSetSkyDomeColorPoint

Syntax

IrrSetSkyDomeColorPoint(dome as irr_node, Red as uinteger, Green as uinteger, Blue as uinteger, pointXPosition as single, pointYPosition as single, pointZPosition as single, pointRadius as single, pointFade as single, addative as uinteger)

Description

Set the color of the verticies in the skydome radiating out from a point. This is powerful effect that can be used to color parts of the skydome and create effects to represent the glows of the rising sun or the moon in the sky. The radius is used to limit the distance of the coloring, pointFade defines the amount that the band is faded into the existing skydome color and addative can be IRR_ON to add the color of the band to the existing color of the skydome or IRR_OFF to replace it.

Example

```
' add a bright golden circle of light at the same point as the
rising sun
IrrSetSkyDomeColorPoint ( SkyDome, 255,220,96, 1000.0, -250.0,
0.0, 1500.0, 0.75, IRR_ON )
```

IrrSetZoneManagerProperties

Syntax

IrrAddZoneManager (zoneManager as irr_node, newNearDistance as single, newFarDistance as single, accumulateChildBoxes as uinteger)

Description

Sets the draw distances of nodes in the zone/distance management node and whether or not the zone manager is to accumulate the bounding boxes of its children as they are added.

Example

IrrSetZoneManagerProperties(zone, 0, 600, IRR_ON)

IrrSetZoneManagerBoundingBox

Syntax

IrrSetZoneManagerBoundingBox (zoneManager as irr_node, x as single, y as single, z as single, boxWidth as single, boxHeight as single, boxDepth as single)

Description

Allows the user to manually set the bounding box of a zone manager node.

Example

IrrSetZoneManagerBoundingBox(zone, 0, 0, 0, 100, 100, 100)

IrrSetZoneManagerAttachTerrain

Syntax

IrrSetZoneManagerAttachTerrain (zoneManager as irr_node, terrain as irr_terrain, structureMapFile as zstring ptr, colorMapFile as zstring ptr, detailMapFile as zstring ptr, ImageX as integer, ImageY as integer, sliceSize as integer)

Description

A special feature of the zone manager is its ability to manage tiled terrain nodes, a zone does this by taking position of an attached terrain object that it shares with other zone objects whenever the camera starts to come into range. The terrain object is loaded with new height information, color and detail ready for when it becomes visible to the camera.

The structureMapFile is the name of an RGBA bitmap file that is to be used to set the structure of the terrain. The Alpha channel is used to set the height and the RGB channels are used to set the color of the vertex at that position. This can be used to load lighting into the scene or to load detail map blending into the scene for simple terrain spattering (discussed in the tile terrain section).

The optional color and detail maps are loaded to apply new color and detail maps to the terrain. If either is not used they should be replaced with IRR_NO_OBJECT.

The Image X and Y define the X and Y position of this tile on the structure and color images, so you could load a 1024x1024 structure image and a 1024x1024 detail image in and have your zones form a grid across these large bitmaps.

Finally SilceSize allows you to only process a slice of the terrain on each frame, as a tile is swapped loading in bitmaps and then adjusting what could be 65,000 vertices in a single frame will cause a noticable hiccup in the smooth running of the scene, so by setting the SliceSize you can define how many rows of the terrain are to be processed on each frame.for example if your tile is 128x128 you might process 32 rows, the tile would then be restructured over 4 frames instead of trying to do it all in one.

Note: You can load your images manually to save them with IrrGetImage and IrrGetTexture and let them stay in memory to avoid having to load images while the scene is running however you should stay aware of how much memory you are using especially the graphics card memory used by IrrGetTexture.

Example

```
IrrSetZoneManagerAttachTerrain ( Zone(X + Y*ROW_SIZE),
Terrain(index), "SunnyValley.tga", "SunnyValley.bmp",
IRR_NO_OBJECT, X*112, Y*112, 32 )
```

IrrSetGrassDensity

Syntax

IrrSetGrassDensity (grass as irr_node, density as integer, distance as single)

Description

Set grass density, density being the number of grass nodes visible in the scene and distance being the distance at which they can be seen.

Example

IrrSetGrassDensity (grassNode, 300, 4000)

IrrSetGrassWind

Syntax

IrrSetGrassWind (grass as irr_node, strength as single, resoloution as single)

Description

Set the grass wind effect, the strength being the strength of the wind, the

resoloution being how often the effect is calculated. By setting the resoloution to zero the wind effect will be stopped and there will be a performance increase however the wind effect adds significantly to the subtle atmosphere of the scene.

Example

IrrSetGrassWind (grassNode, 3.0, 1.0)

IrrGetGrassDrawCount

Syntax

uinteger = IrrGetGrassDrawCount (grass as irr_node)

Description

Get the number of grass objects drawn.

Example

VisibleGrass = IrrGetGrassDrawCount(Grass)

IrrSetFlareScale

Syntax

IrrSetFlareScale (flare as irr_node, source as single, optics as single)

Description

Sets the scale of optics in the scene. The source is the texture used to surround the light source while the options are the scale of textures in the optics of the camera. Sometimes it is effected to make the scale of the source considerably larger than those of the optics and to scale the effect in the optics down so that their appearence is more subtle.

Example

IrrSetFlareScale (FlareNode, 2.0, 1.0)

IrrAddLODManager

Syntax

node = IrrAddLODManager (fadeScale as uinteger = 4, useAlpha as uinteger =

IRR_ON, callback as any ptr = 0)

Description

Adds a level of detail manager to the scene. The primary use for this node is to add other scene nodes to it as children and have their level of detail controlled automatically. If those nodes are made from loaded meshs different meshes containing different amounts of detail can be displayed at different distances.

The other function of the LOD manager is to fade nodes in an out at a specific distance so they gradually fade rather than disappear abruptly. This is achieved by applying a distance without supplying a mesh.

fadeScale is the number of 1/4 seconds that the node takes to fade out or in. 4 units equals 1 second.

useAlpha specifies whether or not the Alpha color of the object is faded too. the callback function is called whenever a node is made invisible or visible. this allows you to stop processing hidden nodes.

Example

LODManager = IrrAddLODManager(4, IRR_ON, @NodeChangeCallback) IrrAddLODMesh(LODManager, 0.0, LOD1Mesh) IrrAddLODMesh(LODManager, 400.0, IRR_N0_OBJECT)

IrrAddLODMesh

Syntax

IrrAddLODMesh (node as irr_node, distance as single, mesh as irr_mesh)

Description

Set the distance at which a particular mesh is to be applied to child mesh nodes. if no mesh is supplied it specifies the distance at which the node should be faded in an out.

node is the LOD manager node

distance is the distance at which this effect will be applied mesh is the mesh used at this distance and beyond or null to specifiy the limit of visibility for this node.

Example

```
LODManager = IrrAddLODMesh( 4, IRR_ON, @NodeChangeCallback )
IrrAddLODMesh( LODManager, 0.0, LOD1Mesh )
IrrAddLODMesh( LODManager, 400.0, IRR_NO_OBJECT )
```

IrrSetLODMaterialMap

Syntax

IrrSetLODMaterialMap (node as irr_node, source as IRR_MATERIAL_TYPES, target as IRR_MATERIAL_TYPES)

Description

Specifies which material is used to apply the fade effect for another material type. How this is used will depend on the effect that you want to achieve. By default fading is applied with the

IRR_EMT_TRANSPARENT_VERTEX_ALPHA material.

node is the LOD manager node

source is the material type your node uses target is the material type used for the fade effect.

Example

IrrSetLODMaterialMap(LODManager, IRR_EMT_TRANSPARENT_ADD_COLOR, IRR_EMT_TRANSPARENT_ADD_COLOR)

IrrAddBillBoardGroupToScene

Syntax

node = IrrAddBillBoardGroupToScene ()

Description

Adds a billboard group to the scene. This is a special object that can have billboard like objects added and removed from it and rendered in a very quick an efficient manner. They are all treated as a single object rather than as many individual nodes. This is particuallarly useful for custom particle effects.

Example

BillboardGroup = IrrAddBillBoardGroupToScene

IrrAddBillBoardToGroup

Syntax

BillboardAddress = IrrAddBillBoardToGroup (
 group as irr_node, _
 sizex as single, sizey as single, _
 x as single = 0, y as single = 0, z as single = 0, _
 roll as single = 0, _
 A as uinteger = 255, R as uinteger = 255, G as uinteger = 255, B as
uinteger = 255)
Description

Adds a billboard to a billboard group. There are a number of properties that are used to specify the billboard.

group is the billboard group node

sizex and sizey are the x and y sizes of the billboard

x, y and z define the position of the billboard

roll specifies the number of degrees that the billboard is spun around its center. A, R, G and B specify the color used for the billboard

Example

```
BillboardAddress = IrrAddBillBoardToGroup( BillboardGroup,_
200.0, 200.0, _
0.0, 0.0, 0.0, _
0.0, _
0, 255, 255, 255 )
```

IrrAddBillBoardByAxisToGroup

```
Syntax
BillboardAddress = IrrAddBillBoardByAxisToGroup (
group as irr_node, __
sizex as single, sizey as single, __
x as single = 0, y as single = 0, z as single = 0, __
roll as single = 0, __
A as uinteger = 255, R as uinteger = 255, G as uinteger = 255, B as
uinteger = 255, __
axis_x as single = 0, axis_y as single = 0, axis_z as single = 0 )
```

Description

Adds a billboard to a billboard group that is fixed to a particular axis these billboards are particularly useful for things like grass. There are a number of properties that are used to specify the billboard.

group is the billboard group node

sizex and sizey are the x and y sizes of the billboard

x, y and z define the position of the billboard

roll specifies the number of degrees that the billboard is spun around its center. A, R, G and B specify the color used for the billboard

axis_x, axis_y, axis_z a direction around which the billboard is spun to face the camera

Example

```
BillboardAddress = IrrAddBillBoardByAxisToGroup( BillboardGroup,_
200.0, 200.0, _
0.0, 0.0, 0.0, _
0.0, _
0, 255, 255, 255, _
0.0, 1.0, 0.0 )
```

IrrRemoveBillBoardFromGroup

Syntax

IrrRemoveBillBoardFromGroup (group as irr_node, billboardAddress as any ptr)

Description

Removes the specified billboard from the billboard group

Example

IrrRemoveBillBoardFromGroup (BillboardGroup, BillboardAddress)

IrrBillBoardGroupShadows

Syntax

Description

Applies lighting to the billboards in a cluster of billboards. This can be used for example to shade the particles in a group of billboards representing a cloud.

group is the group of billboards to which the lighting is to be applied. x, y and z is the direction from which the light is arriving intensity is the strength of the light ambient is the strength of ambient light in the billboard group

Example

IrrBillBoardGroupShadows(BillBoardGroup, 1.0, 0.0, 0.0, 1.0, 0.5

IrrGetBillBoardGroupCount

Syntax

uinteger = IrrGetBillBoardGroupCount (group as irr_node)

Description

Get the number of billboards in the billboard group.

Example

count = IrrGetBillBoardGroupCount (BillboardGroup)

IrrBillBoardForceUpdate

Syntax

IrrBillBoardForceUpdate (group as irr_node)

Description

Unlike regular billboards the billboard group does not always update the orientation of the billboards every frame. If you are a long distance away from the billboard group the camera needs to travel a significant distance before the angle has changed enough to warrent an update of all of the billboards verticies to make them point to the camera once more. You may want to force a refresh at some point with this call.

Example

TrrDillDoordCoroollodoto	1	DillboordCroup	~
IrrBillBoardForceUpdate	(BillboardGroup)

IrrAddBoltSceneNode (Scene Node cr Trivtn)	eated by Sudi with extensions by		
Syntax			
irr_node = IrrAddBoltSceneNode ()			
Description			
The bolt is a special scene node that can	be used to replicate electrical effects.		
This command simply adds the bolt you	-		
properties. This node can be used to simulate lightning and other electrical			
effects.			
Example			
ElectricNode = IrrAddBoltSceneNode()			
IrrSetBoltProperties (SceneNode,	.		
0,90,0, _	' the start point for the bolt		
	' the end point for the bolt ' the bolt updates every 50		
miliseconds	the bort updates every so		
10, _	' the bolt is 10 units wide		
5, _	' the bolt is 5 units thick		
	' there are 10 sub parts in each		
bolt	' there are 4 individual bolts		
, _	' the end is not connected to an		
exact point	the cha is not connected to an		
RGBA(255, 255, 255, 0))	' Lighting color		

IrrSetBoltProperties

Syntax IrrSetBoltProperties (bolt as irr_node, _ startX as single, startY as single, startZ as single, _ endX as single, endY as single, endZ as single, _ updateTime as uinteger = 50, _ radius as uinteger = 10, _ thickness as single = 5.0, _ parts as uinteger = 10, _ bolts as uinteger = 6, _ steadyend as uinteger = IRR_OFF, _ boltColor as uinteger = RGBA(0,0,255,255))

Description

This sets the properties of a bolt node that simulates an electrical effect. There are a number of properties that control many aspects of the bolt to produce a wide range of appearences..

Start X, Y and Z define the point that the bolt origionates from.

End X,Y and Z define the terminating point for the bolt.

Update time specifies the number of miliseconds between updates to the appearence of the bolt.

Radius is the radius of the entire bolt effect.

Thickness is the thickness of a single electrical element in the bolt.

Parts defines the number of segments the bolt is divided into.

Bolts represents the number of individual electrical arcs that are rendered.

SteadyEnd when set to IRR_ON ends in a tight point, when set to IRR_OFF it ends with the same width as the rest of the bolt.

Color specifies the diffuse color that is applied to the bolt.

Color specifies the diffuse color that is ap	
Example	
ElectricNode = IrrAddBoltSceneNode()
IrrSetBoltProperties (SceneNode, _	
0,90,0, _	the start point for the bolt
0,0,0, _	the end point for the bolt
50, _ '	the bolt updates every 50
miliseconds	
10, _ '	the bolt is 10 units wide
5, _ '	the bolt is 5 units thick
10, _ '	there are 10 sub parts in each
bolt	
4, _ '	there are 4 individual bolts
II	

IrrAddBeamSceneNode (Scene Node by Gaz Davidson (Blindside))

Syntax

irr_node = IrrAddBeamSceneNode ()

Description

The beam is a special scene node that can be used to replicate beam effects like lasers and tracer gun fire. This command simply adds the beam you should then make calls to set the beams properties.

Example

BeamNode = IrrAddBeamSceneNode () IrrSetBeamSize (BeamNode, 5.0) IrrSetBeamPosition (BeamNode, X,Y,Z, X+100,Y,Z)

IrrSetBeamSize

Syntax

IrrSetBeamSize (beam as irr_node, size as single)

Description This call sets the width of a beam node

Example

IrrSetBeamSize (BeamNode, 5.0)

IrrSetBeamPosition

Syntax

IrrSetBeamPosition (beam as irr_node, _

startX as single, startY as single, startZ as single, _____ endX as single, endY as single, endZ as single)

Description

This call sets the start and end positions of a beam node. The beam will stretch between the two nodes.

Start X, Y and Z define the point that the bolt origionates from.

End X,Y and Z define the terminating point for the bolt.

Example

IrrSetBeamPosition (BeamNode, X,Y,Z, X+100,Y,Z)

Scene Nodes

Calls for manipulating, texturing and getting information from nodes in the scene.

IrrGetNodeName

Syntax

const zstring ptr = IrrGetNodeName (node as irr_node)

Description Get the name of the node.

Example

NodeName = IrrGetNodeName(StatueNode)

IrrSetNodeName

Syntax

IrrSetNodeName (node as irr_node, name as zstring ptr)

Description Set the name of a node

Example

IrrSetNodeName(StatueNode, "HeroStatue")

IrrGetNodeMesh

Syntax

irr_mesh = IrrGetNodeMesh (node as irr_node)

Description Get the mesh that is associated with a node

Example

myMesh = IrrGetNodeMesh(StatueNode)

IrrGetMaterialCount

Syntax

integer = IrrGetMaterialCount (node as irr_node)

Description

Get the number of materials associated with a node.

Example

nummaterials = IrrGetMaterialCount(StatueNode)

IrrGetMaterial

Syntax

irr_material = IrrGetMaterial(node as irr_node, material_index as integer)

Description

Get the material associated with the node at the particular index

Example

current_material = IrrGetMaterial(StatueNode, index)

IrrSetNodeMaterialTexture

Syntax

IrrSetNodeMaterialTexture(node as irr_node, texture as irr_texture, material_index as integer)

Description

Applys a texture to a node in the scene, how the texture is applied across the surface of the node will depend on the texturing co-ordinates in each of the vectors of the mesh and how they are plotted across the surface of the texture. Some nodes can have several textures applied to them to create special material effects. Node refers to a node you have added to the scene.

Texture refers to a texture you have loaded from an image file.

Material is the index number of the material layer, this will usually be 0 or 1.

Example

IrrSetNodeMaterialTexture(DolphinNode, DolphinTexture, 0)

IrrSetNodeMaterialFlag

Syntax

IrrSetNodeMaterialFlag(node as irr_node, flag as IRR_MATERIAL_TYPES, value as uinteger)

Description

Sets material properies of a node that will effect its appearance on the screen, each of these properties can be either switched on or off.

Node refers to a node that has been added to the scene.

Flag is one of the following properties: -

IRR_EMF_WIREFRAME Render as wireframe outline IRR_EMF_GOURAUD_SHADING Render smoothly across polygons IRR_EMF_LIGHTING Material is effected by lighting IRR_EMF_ZBUFFER Enable z buffer IRR_EMF_ZWRITE_ENABLE Can write as well as read z buffer IRR_EMF_BACK_FACE_CULLING Cull polygons facing away IRR_EMF_BILINEAR_FILTER Enable bilinear filtering IRR_EMF_TRILINEAR_FILTER Enable trilinear filtering IRR_EMF_ANISOTROPIC_FILTER Reduce blur in distant textures IRR_EMF_FOG_ENABLE Enable fogging in the distance IRR_EMF_NORMALIZE_NORMALS Use when scaling dynamically lighted models

The value should be one of the following to switch the property on or off: -IRR_ON IRR_OFF

Example

IrrSetNodeMaterialFlag(CharacterNode, IRR_EMF_GOURAUD_SHADING, IRR_ON)

IrrSetNodeMaterialType

Syntax

IrrSetNodeMaterialType(node as irr_node, mat_type as IRR_MATERIAL_FLAGS)

Description Set the way that materials are applied to the node.

Node refers to a node that has been added to the scene.

Mat_type is one of the following properties that is applied to the node: -

IRR_EMT_SOLID Standard solid rendering uses one texture

IRR_EMT_SOLID_2_LAYER 2 blended textures using vertex alpha value

IRR_EMT_LIGHTMAP 2 textures: 0=color 1=lighting level and ignores vertex lighting

IRR_EMT_LIGHTMAP_ADD

... as above but adds levels instead of modulating between them

IRR_EMT_LIGHTMAP_M2

... as above but color levels are multiplied by 2 for brightening

IRR_EMT_LIGHTMAP_M4 ... as above but color leels are multiplied by 4 for brightening

IRR_EMT_LIGHTMAP_LIGHTING 2 textures: 0=color 1=lighting level but supports dynamic lighting

IRR_EMT_LIGHTMAP_LIGHTING_M2 ... as above but color levels are multiplied by 2 for brightening

IRR_EMT_LIGHTMAP_LIGHTING_M4 ... as above but color levels are multiplied by 4 for brightening

IRR_EMT_DETAIL_MAP 2 blended textures: the first is a color map the second at a different scale adds and subtracts from the color to add detail

IRR_EMT_SPHERE_MAP makes the material appear reflective

IRR_EMT_REFLECTION_2_LAYER a reflective material blended with a color texture

IRR_EMT_TRANSPARENT_ADD_COLOR a transparency effect that simply adds a color texture to the background. the darker the color the more transparent it is.

IRR_EMT_TRANSPARENT_ALPHA_CHANNEL a transparency effect that uses the color textures alpha as a transparency level

IRR_EMT_TRANSPARENT_ALPHA_CHANNEL_REF a transparency effect that uses the color textures alpha, the pixel is only drawn if the alpha is > 127. this is a fast effect that does not blur edges and is ideal for leaves & grass etc.

IRR_EMT_TRANSPARENT_VERTEX_ALPHA a transparency effect that uses the vertex alpha value

IRR_EMT_TRANSPARENT_REFLECTION_2_LAYER a transparent & reflecting effect. the first texture is a reflection map, the second a color map. transparency is from vertex alpha

IRR_EMT_NORMAL_MAP_SOLID

A solid normal map renderer. First texture is color, second is normal map. Only use nodes added with IrrAddStaticMeshForNormalMappingToScene. Only supports nearest two lights. Requires vertex and pixel shaders 1.1

IRR_EMT_NORMAL_MAP_TRANSPARENT_ADD_COLOR ... as above only with a transparency effect that simply adds the color to the background. the darker the color the more transparent it is.

IRR_EMT_NORMAL_MAP_TRANSPARENT_VERTEX_ALPHA ... as above only with a transparency effect that uses the vertex alpha value

IRR_EMT_PARALLAX_MAP_SOLID

similar to the solid normal map but more realistic providing virtual displacement of the surface. Uses the alpha channel of the normal map for height field displacement. Requires vertex shader 1.1 and pixel shader 1.4.

IRR_EMT_PARALLAX_MAP_TRANSPARENT_ADD_COLOR ... as above only with a transparency effect that simply adds the color to the background. the darker the color the more transparent it is.

IRR_EMT_PARALLAX_MAP_TRANSPARENT_VERTEX_ALPHA ... as above only wiht a transparency effect that uses the vertex alpha value

Example

IrrSetNodeMaterialType(WaterNode, IRR_EMT_LIGHTMAP)

IrrSetNodePosition

Syntax

IrrSetNodePosition(node as irr_node, X as single, Y as single, Z as single)

Description Moves the node to the new position.

Example

IrrSetNodePosition(CharacterNode, 500.0, 100.7, -192.6)

IrrSetNodeRotation

Syntax

IrrSetNodeRotation(node as irr_node, X as single, Y as single, Z as single)

Description

Rotate a node to the specified orientaion through its X, Y and Z axis

Example

IrrSetNodeRotation(CharacterNode, 34.5 0.76, -67.3)

IrrSetNodeScale

Syntax

IrrSetNodeScale(node as irr_node, X as single, Y as single, Z as single)

Description

Change the scale of a node in the scene making it bigger or smaller in the X, Y and Z axis

Example

IrrSetNodeScale(CharacterNode, 1.2,1.5,1.2)

IrrSetNodeRotationPositionChange

Syntax

IrrSetNodeRotationPositionChange(node as irr_node, roll as single, pitch as single, yaw as single, drive as single, strafe as single, elevate as single, forwardVector as IRR_VECTOR ptr, upVector as IRR_VECTOR ptr, offsetVectorCount as integer, offsetVectors as IRR_VECTOR ptr)

Description

Apply a change in rotation and a directional force. we can also optionally recover pointers to a series of vectors. The first is a pointer to a vector pointing forwards the second is a pointer a vector pointing upwards following this are any number of points that will also be rotated (the effect on these points is NOT accumulative so the points should be initialised with their origonal values each time this is called)

Example

```
IrrSetNodeRotationPositionChange( SceneNode, roll, pitch, yaw,
drive, strafe, elevate, @forwardVector, @upVector, 2,
@cameraVector(0))
```

IrrDebugDataVisible

Syntax

IrrDebugDataVisible (node as irr_node, visible as integer)

Description

Displays debugging data around a node, this typically means drawing the bounding box around the edges of the node.

There are a series of values for displaying different types of debugging information and not all of them are supported on all node types

0 No Debugging 1 Bounding Box 2 Normals 4 Skeleton 8 Wireframe 16 Transparency 32 Bounding Box Buffers &hffffffff Everything

Example

IrrDebugDataVisible (PyramidNode, 1)

IrrGetNodePosition

Syntax

IrrGetNodePosition(node as irr_node, X as single, Y as single, Z as single)

Description

Gets the position of a node in the scene and stores its X, Y and Z co-ordinates into the supplied variables.

Example

IrrGetNodePosition(CharacterNode, XPosition, YPosition, ZPosition
)

IrrGetNodeAbsolutePosition

Syntax

IrrGetNodeAbsolutePosition(node as irr_node, X as single, Y as single, Z as single)

Description

Get the absoloute position of the node in the scene this position includes the position changes of all of the nodes parents too.

Example

IrrGetNodeAbsolutePosition(CharacterNode, XPosition, YPosition, ZPosition)

IrrGetNodeRotation

Syntax

IrrGetNodeRotation(node as irr_node, X as single, Y as single, Z as single)

Description

Get the rotation of a node in the scene and stores the X, Y and Z rotation values in the supplied variables..

Example

IrrGetNodeRotation(CharacterNode, XRotation, YRotation, ZRotation

IrrGetNodeScale

Syntax

IrrGetNodeScale(node as irr_node, X as single, Y as single, Z as single)

Description

Get the scale of a node in the scene and stores the X, Y and Z scale values in the supplied variables..

Example

IrrGetNodeScale(CharacterNode, XScale, YScale, ZScale)

IrrGetJointNode

Syntax

irr_node = IrrGetJointNode (node as irr_node, joint_name as zstring ptr)

Description

This supplies you with an invisible node that follows the motion of a particular joint in an animated models skeleton. You can use this to attach child nodes that represent objects a person is carrying for example. (This call now replaces IrrGetMS3DJointNode and IrrGetDirectXJointNode which are only supplied for backwards compatibility). It can now also be used to manually move the joint.

The name should refer to the name of a joint in the model.

Example

IrrAddChildToParent

Syntax

IrrAddChildToParent (child as irr_node, parent as irr_node)

Description

Attaches the child node to the parent node, whenever you change the parent node the child node changes too. This is useful for putting a cup in a characters hand for example. You can move and rotate the child node to move the object into position against its parent.

Example

IrrAddChildToParent(CupNode, CharacterNode)

IrrGetNodeFirstChild

Syntax

irr_node = IrrGetNodeFirstChild (node as irr_node, searchPosition as any ptr)

Description

Get the first child node of this node, returns 0 if there is no child.

Example

ChildNode = IrrGetNodeFirstChild (SectorNode, position)

IrrGetNodeNextChild

Syntax

irr_node = IrrGetNodeNextChild (node as irr_node, searchPosition as any ptr)

Description

Get the next child node of this node, returns 0 if there is no child.

Example

ChildNode = IrrGetNodeNextChild(SectorNode, position)

IrrIsNodeLastChild

Syntax

integer = IrrIsNodeLastChild (child as irr_node, parent as irr_node)

Description

Attaches the child node to the parent node, whenever you change the parent node the child node changes too. This is useful for putting a cup in a characters hand for example. You can move and rotate the child node to move the object into position against its parent. Example

```
if IrrIsNodeLastChild( SectorNode, position ) = 0 then
    LastNode = IRR_YES
end if
```

IrrAddNodeShadow

Syntax

irr_node = IrrAddNodeShadow (node as irr_node, mesh as irr_mesh = 0)

Description

Adds shadows to a node that are cast across other nodes in the scene, shadowing need to be turned on when you call IrrStart. You should analyse the performance of your scene carefully when using this function as it can have a significant effect on your frame rate. You can supply a different mesh to the one used to display the node, this shadow mesh could be a much lower resoloution than that used for your model thereby improving performance.

Example

IrrAddNodeShadow (CharacterNode)

IrrSetNodeVisibility

Syntax

IrrSetNodeVisibility (node as irr_node, visible as integer)

Description

This allows you to hide nodes from the display so you can quickly and easily switch objects out to improve performance or create effects liek one node transforming into another node (perhaps in a puff of particle smoke).

Visible can be one of the following values: -IRR_INVISIBLE IRR_VISIBLE

Example

IrrSetNodeVisibility(CharacterNode, IRR_VISIBLE)

IrrRemoveNode

Syntax IrrRemoveNode(node as irr_node)

Description

Removes a node from the scene deleting it.

Example

IrrRemoveNode(CharacterNode)

Syntax IrrRemoveAllNodes()

Description

Clears the entire scene, any references to nodes in the scene will become invalid.

Example

IrrRemoveAllNodes()

IrrSetNodeParent

Syntax

IrrSetNodeParent (node as irr_node, parent as irr_node)

Description Sets the parent of the specified node.

Example

ParentNode = IrrSetNodeParent(ChildNode, ParentNode)

IrrGetNodeParent

Syntax

irr_node = IrrGetNodeParent (node as irr_node)

Description

Gets the parent of the specified node.

Example

ParentNode = IrrGetNodeParent(ChildNode)

IrrGetNodeID

Syntax integer = IrrGetNodeID (node as irr_node)

Description

Each node can have a 32 bit signed identification number assigned to them this can be used in collision operations to filter out particular classes of object.

Example

NodeID = IrrGetNodeID(TreeNode)

IrrSetNodeID

Syntax

IrrSetNodeID (node as irr_node, id as integer)

Description

Adds a simple cube object to the scene. This is particularly useful for testing and is a quick and easy way of playing objects into the scene for testing placement.

Example

IrrSetNodeID (TreeNode, 8)

IrrGetNodeBoundingBox

Syntax

integer = IrrGetNodeBoundingBox (node as irr_node, x1 as single, y1 as single, z1 as single, x2 as single, y2 as single, z2 as single,)

Description

Gets the coordiantes describing the bounding box of the node into the six supplied variables.

Example

NodeID = IrrGetNodeBoundingBox(BuildingNode, Xa, Ya, Za, Xb, Yb, Zb)

IrrGetNodeTransformedBoundingBox

Syntax

integer = IrrGetNodeTransformedBoundingBox (node as irr_node, x1 as single, y1 as single, z1 as single, x2 as single, y2 as single, z2 as single,)

Description

Gets the transformed (absolute value) bounding box of a node into the six supplied variables. So if your node has been moved hundreds of units away from the origion the co-ordinates of its bounding box will also be hundreds of units away corisponding to its real location in the scene.

Example

NodeID = IrrGetNodeTransformedBoundingBox(BuildingNode, Xa, Ya, Za, Xb, Yb, Zb)

Animation

Calls that control the animation of nodes in the scene either by playing animation that is embedded in the mesh or applying animator controls to automatically effect the nodes.

IrrSetNodeAnimationRange
Syntax IrrSetNodeAnimationRange(node as irr_node, Start Frame as integer, End Frame as integer)
Description Sets the range of animation that is to be played in the node. An anaimation sequences might run from 0 to 200 frames and a sequence where your character is running might only occupy a portion of this.
Example
IrrSetNodeAnimationRange(CharacterNode, 50, 75)

IrrPlayNodeMD2Animation

Syntax

IrrPlayNodeMD2Animation (node as irr_node, sequence as uinteger)

Description

MD2 format models have specific animation sequences contained within them that can be played back with a simple call.

sequence should be one of the following values: -IRR_EMAT_STAND IRR_EMAT_RUN IRR_EMAT_ATTACK IRR EMAT PAIN A IRR EMAT PAIN B IRR_EMAT_PAIN_C IRR_EMAT_JUMP IRR_EMAT_FLIP IRR_EMAT_SALUTE IRR_EMAT_FALLBACK IRR_EMAT_WAVE IRR_EMAT_POINT IRR_EMAT_CROUCH_STAND IRR_EMAT_CROUCH_WALK IRR EMAT CROUCH ATTACK IRR_EMAT_CROUCH_PAIN IRR_EMAT_CROUCH_DEATH IRR EMAT DEATH FALLBACK IRR EMAT DEATH FALLFORWARD IRR EMAT DEATH FALLBACKSLOW IRR_EMAT_BOOM

Example

IrrPlayNodeMD2Animation(CharacterNode, IRR_EMAT_STAND)

IrrSetNodeAnimationSpeed

Syntax

IrrSetNodeAnimationSpeed (node as irr_node, speed as integer)

Description

Change the speed at which an animation is played for a node. You could use this to make a character run slowly or quickly and still keep its feet on the ground.

Example

IrrSetNodeAnimationSpeed(CharacterNode, 25)

IrrGetNodeAnimationFrame

Syntax

uinteger = IrrGetNodeAnimationFrame(node as irr_node)

Description

Get the frame number that is currently being played by the node.

Example

CurrentFrame = IrrGetNodeAnimationFrame(AnimNode)

IrrSetNodeAnimationFrame

Syntax

IrrSetNodeAnimationFrame(node as irr_node, frame as integer)

Description

Set the current frame number being played in the animation

Example

IrrSetNodeAnimationFrame(CharacterNode, 75)

IrrSetTransitionTime

Syntax

IrrSetTransitionTime (node as irr_node, speed as single)

Description

Sets the transition time across which two poses of an animated mesh are blended. For example a character in a sitting pose can be switched into a lying down pose by blending the two frames, this will provide a more convincing smooth transition instead of a snap change in position. IrrAnimateJoints must be called before IrrDrawScene if blending is used.

Example

IrrSetTransitionTime(CharacterNode, 0.75)

IrrAnimateJoints

Syntax IrrAnimateJoints(node as irr_node)

Description

Animates the mesh based on the position of the joints, this should be used at the end of any manual joint operations including blending and joints animated using IRR_JOINT_MODE_CONTROL and IrrSetNodeRotation on a bone node.

Example

IrrAnimateJoints(CharacterNode)

IrrSetJointMode

Syntax

IrrSetJointMode (node as irr_node, mode as uinteger)

Description

Sets the animation mode of joints in a node. When using the control mode IrrAnimateJoints must be called before IrrDrawScene.

IRR_JOINT_MODE_NONE will result in no animation of the model based on bones

IRR_JOINT_MODE_READ will result in automatic animation based upon the animation defined with calls like IrrSetNodeAnimationRange

IRR_JOINT_MODE_CONTROL will allow the position of the bones to be set through code

Example

IrrSetJointMode(CharacterNode, IRR_JOINT_MODE_CONTROL)

IrrAddCollisionAnimator

Syntax

irr_animator = IrrAddCollisionAnimator (selector as irr_selector, node as irr_node, radius x as single, radius y as single, radius z as single, gravity x as single, gravity y as single, gravity z as single, offset x as single, offset y as single, offset z as single)

Description

This adds a collision animator to a node that applies collision detection and gravity to the object. The collision detection will stop the object penetrating through a surface in the objects it is colliding against and will also press it against the surface using gravity.

Selector represents a selection of triangles in the scene, this is usually all of the triangles in a map for instance. Please refer to the section on collision for further details of how to obtain this object.

Radius X, Radius Y and Radius Z define an ellipsoid that defines the area of collision this eliptical shape allows the collision detection to slide the object up steps and even ladders. If you make it too big you might be too large to get through a doorway but if you make it too small you may not be able to climb steps. You should play with these values and find the best ones for your scene.

Gravity X, Gravity Y and Gravity Z work together to specify the force that is applied to the node to make it drop to the ground. Other values could be used to simulate wind effects.

Offset X, Offset Y and Offset Z are used to offset the node by a specific distance from the center of the collision, as the center of the object and the size of your collision ellipsoid vary you can use this to adjust the position of the node and to bring it into contact with the ground.

Example

CollisionAnimator = IrrAddCollisionAnimator(MapCollision, CameraNode, 30.0,30.0,30.0, 0.0,-3.0,0.0, 0.0,50.0,0.0)

IrrAddDeleteAnimator

Syntax

irr_animator = IrrAddDeleteAnimator (node as irr_node, milliseconds to deletion as integer)

Description

This animator deletes the node it is attached to after the specified number of milliseconds (1/1000ths of a second). You could use this animator to delete a falling rock for example, all you would need to do is attach the delete animator, a movement animator and then forget about it.

Example

DeleteAnimator = IrrAddDeleteAnimator(RockNode, 3000)

IrrAddFadeAnimator

Syntax

irr_animator = IrrAddFadeAnimator (node as irr_node, milliseconds to deletion as integer, scale as single)

Description

This animator deletes the node it is attached to after the specified number of milliseconds (1/1000ths of a second). During the time while it is waiting to delete it the node is slowly faded to invisibility and is also scaled by the specified amount. You could use this animator to fade and delete an object from a scene that was no longer required like a used medical pack, all you would need to do is attach the fade animator and forget about it.

Example

FadeAnimator = IrrAddFadeAnimator(MedicalNode, 3000, 0.0)

IrrAddFlyCircleAnimator

Syntax

irr_animator = IrrAddFlyCircleAnimator (node as irr_node, center x as single, center y as single, center z as single, radius as single, speed as single)

Description

This animator moves the node it is attached to in a circular path.

Center X, Center Y and Center Z define the center of the circular path.

Radius defines the radius of the path

Speed defines the rate the node moves around the circular path

Example

CircleAnimator = IrrAddFlyCircleAnimator(PowerNode, 0,0,0 50, 20

)

Syntax

IrrAddFlyStraightAnimator

Description This animator makes the node it is attached to move in a straight line from the start to the end end point. It would be useful for objects moving on a conveyor belt for example Start X, Start Y and Start Z specify the start point of the path. End X, End Y and End Z specify the end point of the path. Time to complete specifies the number of milliseconds the animator will take to move the node from the start to the end point Loop path determines if the node will be moved from the start to the end and then stopped or whether the animation will be looped this parameter should be either: -IRR ONE SHOT For a single animation and then stop IRR LOOP To continuously repeat the animation Example FlyAnimator = IrrAddFlyStraightAnimator(AnimatedBox, 0,50,-300, 0,50,300, 3000, IRR_LOOP)

irr_animator = IrrAddFlyStraightAnimator (node as irr_node, start x as

as single, time to complete as uinteger, loop path as integer)

single, start y as single, start z as single, end x as single, end y as single, end z

IrrAddRotationAnimator

Syntax

irr_animator = IrrAddRotationAnimator (node as irr_node, x as single, y as single, z as single) Description

This animator makes the node it is attached to spin around.

X, Y and Z specify the number of radians the object is spun around each axis

Example

IrrAddSplineAnimator

Syntax

irr_animator = IrrAddSplineAnimator (node as irr_node, array size as integer, x as single, y as single, z as single, time to start as integer, speed as single, tightness as single)

Description

This is one of the more difficult to set up of the animators but is very natural looking and powerful. A spline is a curved line that passes through or close to a list of co-ordinates, creating a smooth flight. This animator needs a list of coordinates stored in three arrays, one array each for the X, Y and Z locations of all the points. A good way to get co-ordinates for these arrays is to load in the camera position example program and move your camera to a point and write down its co-ordinates.

Array size specifies how many points there are in your spline motion.

The three arrays X, Y and Z containing co-ordinates are passed by reference as a pointer therefore you should ensure that the array is the correct size otherwise unpredictable results will be obtained.

Time to start specifies the number of milliseconds that must pass before the animation starts.

Speed defines the rate the node moves along the spline curve.

Tightness specifies how tightly the curve is tied to the points (0 is angular and 1 is very loose)

Example

-		
		SplineY((0) = 50 : SplineZ((0) = 0
		SplineY((1) = 100 : SplineZ((1) = -100
SplineX(2) =	100 :	SplineY((2) = 50 : SplineZ((2) = 0)
SplineX(3) =	· 0:	SplineY((3) = 100 : SplineZ((3) = 100)
SplineAnimat	or = Ir	rAddSplineAnimator(CameraNode, 4,
<pre>SplineX(0),</pre>	SplineY	(0), SplineZ(0), 0, 0.5, 1)

IrrRemoveAnimator

Syntax

IrrRemoveAnimator (node as irr_node, node as irr_animator)

Description

This removes an animator from a node. Stopping the animation or cleaning an animator up so you can apply a new one.

Example

IrrRemoveAnimator(DoorNode, FlyAnimator)

Collision

Calls for creating collision groups and for calculating collisions in the scene.

IrrGetCo	lisionGroupFromMe	hae
Interco	nsionGrouperonning	2511

Syntax

irr_selector = IrrGetCollisionGroupFromMesh (mesh as irr_mesh, node as irr_node)

Description

Creates a collision object from the triangles contained within the specified mesh as applied to the position, rotation and scale of the supplied node.

Example

ObjectSelector = IrrGetCollisionGroupFromMesh(SimpleBuildingMesh, MyBuilding)

IrrGetCollisionGroupFromComplexMesh

Syntax

irr_selector = IrrGetCollisionGroupFromComplexMesh (mesh as irr_mesh, node as irr_node)

Description

Creates an optimized triangle selection group from a large complex mesh like a map. This group can then be used in collision functions to collide objects against this node. You need to supply both the mesh the node was created from and the node itself.

Example

MapSelector = IrrGetCollisionGroupFromComplexMesh(MapMesh, MapNode)

IrrGetCollisionGroupFromBox

Syntax

irr_selector = IrrGetCollisionGroupFromBox (node as irr_node)

Description

Creates a collision object from the bounding box of a node.

Example

IrrGetCollisionGroupFromTerrain

Syntax

irr_selector = IrrGetCollisionGroupFromTerrain (node as irr_node, level of detail as integer)

Description

Creates a collision object from a terrain node. A higher level of detail improves the collision detection but consumes more resources and can effect the speed of the process.

Example

TerrainSelector = IrrGetCollisionGroupFromTerrain(TerrainNode, 1

IrrRemoveCollisionGroup

Syntax

IrrRemoveCollisionGroup (collisionGroup as irr_selector, node as irr_node)

Description

Remove the collision selector from memory. This collision selector must not be attached to another collision group when it is removed, the collision group is first removed from the node you supply.

Example

IrrRemoveCollisionGroup(buildingCollision, buildingNode)

IrrAttachCollisionGroupToNode

Syntax

IrrAttachCollisionGroupToNode (collisionGroup as irr_selector, node as irr_node)

Description

Attaches a collision group that you have already created from a mesh and a node to another node without duplicating the collision geometry.

Example

IrrAttachCollisionGroupToNode(boxCollision, anotherBoxNode)

IrrSetNodeTriangleSelector

Syntax

IrrSetNodeTriangleSelector (node as irr_node, collisionGroup as irr_selector)

Description

Assigns a collision group to a specific node..

Example

IrrSetNodeTriangleSelector(newBuilding, buildingCollision)

IrrCreateCombinedCollisionGroup

Syntax

irr_selector = IrrCreateCombinedCollisionGroup

Description

Creates a collision object that can be used to combine several collision objects together so you could add a couple of maps and a terrain for example. Initially the combined collision object is empty.

Example

SelectorGroup = IrrCreateCombinedCollisionGroup

IrrAddCollisionGroupToCombination

Syntax

IrrAddCollisionGroupToCombination (combined collision group as irr_selector, collision group as irr_selector)

Description

Adds a collision object to group of collision objects.

Example

```
IrrAddCollisionGroupToCombination( SelectorGroup, MapSelector )
IrrAddCollisionGroupToCombination( SelectorGroup, TerrainSelector
)
```

IrrRemoveAllCollisionGroupsFromCombination

Syntax

IrrRemoveAllCollisionGroupsFromCombination (combined collision group as irr_selector)

Description

Empty a collision group object so that you can add different collision groups to it.

Example

IrrRemoveAllCollisionGroupsFromCombination(SelectorGroup)

IrrRemoveCollisionGroupFromCombination

Syntax

IrrRemoveCollisionGroupFromCombination (combined collision group as irr_selector, collision group as irr_selector)

Description

Remove a single specified collision object from a group of collision objects.

Example

IrrRemoveCollisionGroupFromCombination(SelectorGroup, TerrainSelector)

IrrGetCollisionPoint

Syntax

integer = IrrGetCollisionPoint (start as IRR_VECTOR, line_end as IRR_VECTOR, collision group as irr_selector, collision point as IRR_VECTOR)

Description

Detect the collision point of a ray in the scene with a collision object if a collision was detected 1 is returned and vector collision contains the coordinates of the point of collision

Start defines the start point of the ray and End defines the endpoint

Collision group is a selector object created with one of the above functions.

Collision point is the co-ordinates in 3D space of the collision object the ray and the selector object.

Example

collided = IrrGetCollisionPoint (StartVector, EndVector, CharacterSelector, CollisionVector)

IrrGetRayFromScreenCoordinates

Syntax

IrrGetRayFromScreenCoordinates (screen x as integer, screen y as integer, camera as irr_camera, ray start as IRR_VECTOR, ray end as IRR_VECTOR)

Description

Gets a ray that goes from the specified camera and through the screen coordinates the information is copied into the supplied start and end vectors. You can then use this ray in other collision operations.

Example

```
IrrGetRayFromScreenCoordinates ( screen_x, screen_y, CameraNode,
StartVector, EndVector )
```

IrrGetCollisionNodeFromCamera

Syntax

irr_node = IrrGetCollisionNodeFromCamera (camera as irr_camera)

Description

A ray is cast through the camera and the nearest node that is hit by the ray is returned. If no node is hit zero is returned for the object

Example

TargetedNode = IrrGetCollisionNodeFromCamera (CameraNode)

IrrGetCollisionNodeFromRay

Syntax

irr_node = IrrGetCollisionNodeFromRay (start as IRR_VECTOR, line_end as IRR_VECTOR)

Description

A ray is cast through the supplied coordinates and the nearest node that is hit by the ray is returned. If no node is hit zero is returned for the object

Example

IrrGetCollisionNodeFromScreenCoordinates

Syntax

irr_node = IrrGetCollisionNodeFromScreenCoordinates (screen x as integer, screen y as integer)

Description

A ray is cast through the screen at the specified co-ordinates and the nearest node that is hit by the ray is returned. If no node is hit zero is returned for the object.

Example

```
SelectedNode = IrrGetCollisionNodeFromScreenCoordinates( MouseX,
MouseY )
```

IrrGetScreenCoordinatesFrom3DPosition

Syntax

IrrGetScreenCoordinatesFrom3DPosition (screen x as integer, screen y as integer, at position as IRR_VECTOR)

Description

Screen co-ordinates are returned for the position of the specified 3D coordinates as if an object were drawn at them on the screen, this is ideal for drawing 2D bitmaps or text around or on your 3D object on the screen for example in the HUD of an aircraft. After the call Screen X and Screen Y will contain the co-ordinates.

Example

IrrGetScreenCoordinatesFrom3DPosition (XPosition, YPosition, RocketVector)

IrrGet2DPositionFromScreenCoordinates (contributed by agamemnus)

Syntax

IrrGet2DPositionFromScreenCoordinates (screenx As integer, screeny As integer, x As Single, y As Single, camera As irr_camera)

Description

Calculates the intersection between a ray projected through the specified screen co-ordinates and a plane at the world origin.

The Parameters X, Y and Z will recieve the 2D position where the line through the screen intersects with the plane.

Example

```
IrrGet2DPositionFromScreenCoordinates ( 256, 256, x, y, OurCamera
```

, IrrSetNodePosition(MyCursor, XPosition, YPosition, ZPosition)

IrrGet3DPositionFromScreenCoordinates (contributed by agamemnus)

Syntax

IrrGet3DPositionFromScreenCoordinates (screenx as integer, screeny as integer, x as single, y as single, z as single, camera as irr_camera, normalX as single = 0.0, normalY as single = 0.0, normalZ as single = 1.0, distanceFromOrigin as single = 0.0)

Description

Calculates the intersection between a ray projected through the specified screen co-ordinates and a plane defined from a normal and the distance of that plane from the world origin.

The Parameters X, Y and Z will recieve the 3D position where the line through the screen intersects with the plane.

Example

```
IrrGet3DPositionFromScreenCoordinates ( ScreenX, ScreenY,
XPosition, YPosition, ZPosition, MyCamera )
IrrSetNodePosition( MyModel, XPosition, YPosition, ZPosition )
```

IrrGetChildCollisionNodeFromRay

Syntax

irr_node = IrrGetChildCollisionNodeFromRay (node as irr_node, idMask as integer, recurse as uinteger, start as IRR_VECTOR, line_end as IRR_VECTOR

Description

A ray is cast through the supplied coordinates and the nearest node that is hit by the ray is returned. if no node is hit zero is returned for the object, only a subset of objects are tested, i.e. the children of the supplied node that match the supplied id.Iif the recurse option is enabled the entire tree of child objects connected to this node are tested.

Example

```
IrrGetChildCollisionNodeFromRay ( SectorNode, 100, IRR_OFF,
StartPoint, EndPoint )
```

IrrGetChildCollisionNodeFromPoint

Syntax

irr_node = IrrGetChildCollisionNodeFromPoint (node as irr_node, idMask as integer, recurse as uinteger, point as IRR_VECTOR)

Description

The node and its children are recursively tested and the first node that contains the matched point is returned. if no node is hit zero is returned for the object, only a subset of objects are tested, i.e. the children of the supplied node that match the supplied id. if the recurse option is enabled the entire tree of child objects connected to this node are tested.

Example

```
IrrGetChildCollisionNodeFromPoint ( SectorNode, 100, IRR_ON,
TestPoint )
```

IrrGetNodeAndCollisionPointFromRay

Syntax

irr_node = IrrGetNodeAndCollisionPointFromRay (vectorStart as IRR_VECTOR, vectorEnd as IRR_VECTOR, node as irr_node, posX as single, posY as single, posZ as single, normalX as single, normalY as single, normalZ as single, id as integer = 0, rootNode as irr_node = IRR_NO_OBJECT)

Description

A ray is cast through the specified co-ordinates and the nearest node that has a collision selector object that is hit by the ray is returned along with the coordinate of the collision and the normal of the triangle that is hit. if no node is hit zero is returned for the object. If a node is supplied for the rootNode that tests for collision start from that node and are only tested against that node and its children.

Example

IrrGetRayFromScreenCoordinates (screen_x, screen_y, CameraNode, StartVector, EndVector) IrrGetNodeAndCollisionPointFromRay (StartVector, EndVector,

```
collidedNode, hitX, hitY, hitZ, normalX, normalY, normalZ, 0,
myRoom )
if NOT collidedNode = IRR_NO_OBJECT then
        Print "We hit something"
end if
```

IrrGetDistanceBetweenNodes

Syntax

distance = IrrGetDistanceBetweenNodes (nodeA as IRR_NODE, nodeA as IRR_NODE)

Description

The distance between two nodes is measured using fast maths functions that will show inaccuracies. Useful for when it is nessecary to test distances between many nodes..

Example

Dim As Single Distance = IrrGetDistanceBetweenNodes(nodeA, nodeB
)

IrrAreNodesIntersecting

Syntax

test = IrrAreNodesIntersecting (nodeA as IRR_NODE, nodeA as IRR_NODE

Description

Tests whether the bounding boxes are two nodes are intersecting. Bounding boxes are axis aligned and do not rotate when you rotate the nodes. This should be kept in mind when testing for collisions.

Example

```
If NOT IrrAreNodesIntersecting ( nodeA, nodeB ) = 0 Then
Print "Collision"
End If
```

IrrIsPointInsideNode

Syntax

irr_node = IrrIsPointInsideNode (node as IRR_NODE, X as Single, Y as Single, Z as Single)

Description

Determine if the specified point is inside the bounding box of the node.

Example

```
If NOT IrrIsPointInsideNode ( node, X, Y, Z ) = 0 Then
        Print "Point is inside Node"
End If
```

IrrGetCollisionResultPosition (contributed by The Car)

Syntax

IrrGetCollisionResultPosition (selector As irr_selector, ellipsoidPosition As IRR_VECTOR, ellipsoidRadius As IRR_VECTOR, velocity As IRR_VECTOR, gravity As IRR_VECTOR, slidingSpeed as single, outPosition As IRR_VECTOR, outHitPosition As IRR_VECTOR, outFalling As Integer)

Description

Collides a moving ellipsoid with a 3d world with gravity and returns the resulting new position of the ellipsoid, the point at which the elipsoid collided with the surface and whether the ellipsoid is falling through the air.

This can be used for moving a character in a 3d world: The character will slide at walls and is able to walk up stairs. The method used how to calculate the collision result position is based on the paper "Improved Collision detection and Response" by Kasper Fauerby.

Example

vectPosition.Y, vectPosition.Z)

_

Cameras

Calls for creating and controlling cameras in the scene. The camera objects are used for defining a view point and a target point which is used to render the scene.

IrrAddFPSCamera

Syntax irr_camera = IrrAddFPSCamera

Description

Adds a 'first person shooter' style camera into the scene that will be used to define the view point and target point and other attributes of the view into the 3D scene. If you haven't captured mouse and keyboard events this camera can be controlled with the cursor keys and the mouse.

Example

FPSCamera = IrrAddFPSCamera

IrrAddCamera

Syntax

irr_camera = IrrAddCamera (camera X as single, camera Y as single, camera Z as single, target X as single, target Y as single, target Z as single)

Description

Adds a camera to into the scene that will be used to define the view point and target point and other attributes of the view into the 3D scene. Animators and other node functions can be applied to this node.

Camera X, Camera Y and Camera Z define the view point of the camera.

Target X, Target Y and Target Z define the target of the camera,

Example

CameraObject = IrrAddCamera(100,0,0, 0,-10,0)

IrrAddMayaCamera

Syntax

irr_camera = IrrAddMayaCamera (parent as irr_node, rotateSpeed as single, zoomSpeed as single, moveSpeed as single)

Description

Adds a Maya style camera to into the scene the user can click with the left, middle and right mouse buttons to move, zoom and rotate the camera.

rotateSpeed the speed at which the camera revolves zoomSpeed the speed at which the camera zooms in and out moveSpeed the speed at which the camera moves

Example

CameraObject = IrrAddMayaCamera(IRR_NO_OBJECT, 100.0, 100.0, 100.0,

IrrSetCameraTarget

Syntax

IrrSetCameraTarget (camera as irr_camera, X as single, Y as single, Z as single

Description

The camera view point can be moved by simply using the IrrSetNodePosition function but this operation will change the point that the camera is pointing at.

Example

IrrSetCameraTarget (CameraObject, 0, 50, 0)

IrrGetCameraTarget

Syntax

IrrGetCameraTarget (camera as irr_camera, X as single, Y as single, Z as

single)

Description

Get the point in space that the camera is looking at. The point is copied into the supplied X, Y and Z variables

Example

IrrGetCameraTarget (CameraObject, LookAtX, LookAtY, LookAtZ)

IrrGetCameraUpDirection

Syntax

IrrGetCameraUpDirection (camera as irr_camera, X as single, Y as single, Z as single)

Description

Get the up vector of a camera object into the supplied variables, this controls the upward direction of the camera and allows you to roll it for free flight action. This specifies a point in space at which the top of the camera points.

Example

IrrGetCameraUpDirection (CameraObject, TopOfCamPointsAtX, TopOfCamPointsAtY, TopOfCamPointsAtZ)

IrrSetCameraUpDirection

Syntax

IrrSetCameraUpDirection (camera as irr_camera, X as single, Y as single, Z as single)

Description

Set the up vector of a camera object, this controls the upward direction of the camera and allows you to roll it for free flight action. This specifies a point in space at which the top of the camera points.

Example

```
IrrSetCameraUpDirection ( CameraObject,
TopOfCamPointsAtX, TopOfCamPointsAtY, TopOfCamPointsAtZ )
```

IrrGetCameraOrientation

Syntax

IrrGetCameraOrientation (camera as irr_camera, X as IRR_VECTOR, Y as IRR_VECTOR, Z as IRR_VECTOR)

Description

Gets the vectors describing the camera direction useful after the camera has been revolved.

Example

IrrGetCameraOrientation (CameraObject, VectorX, VectorY, VectorZ)

IrrRevolveCamera

Syntax

IrrRevolveCamera (camera as irr_camera, yaw as single, pitch as single, roll as single, drive as single, strafe as single, elevate as single)

Description

Revolve the camera using quaternion calculations, this will help avoid gimbal lock associated with normal Rotations and is ideal for spacecraft and aircraft.

The command takes six parameters that control yaw (turning left and right), pitch (tilting up and down), roll (rolling left and right), drive (moving forwards and backward), strafe (moving left and right) and finally elevate (moving up and down)

Many thanks to RogerBorg for this.

Example

```
IrrRevolveCamera ( CameraObject, CameraYaw, CameraPitch,
CameraRoll, CameraDrive, CameraDrive, CameraStrafe, CameraElevate
)
```

IrrSetCameraUpAtRightAngle

Syntax IrrSetCameraUpAtRightAngle (camera as irr_camera)

Description

Set the camera up at a right angle to the camera vector.

Example

IrrSetCameraUpAtRightAngle (CameraObject)

IrrSetCameraOrthagonal

Syntax

IrrSetCameraOrthagonal (camera as irr_camera, distanceX as single, distanceY as single, distanceY as single)

Description

Set the projection of the camera to an orthagonal view, where there is no sense of perspective. The distance to the target adjusts the width and height of the camera view, essentially the smaller it is the larger the object will appear.

Example

```
IrrGetNodePosition( MyTarget, tarX, tarY, tarZ )
IrrGetNodePosition( MyCamera, camX, camY, camZ )
IrrSetCameraOrthagonal ( MyCamera, camX-tarX, camY-tarY, camZ-tarZ
)
```

IrrSetCameraClipDistance

Syntax

IrrSetCameraClipDistance (camera as irr_camera, distance as single)

Description

A camera clips objects in the distance that may be a part of the scene to increase rendering performance without requiring you to manage adding and deleting the objects from the view. This defines the distance beyond which no polygons will be drawn.

Example

IrrSetCameraClipDistance (CameraObject, 12000)

IrrSetActiveCamera

Syntax IrrSetActiveCamera (camera as irr_camera)

Description

When you have several camera objects in the scene you can use this call to define which of them is to be used to look through when drawing the scene.

Example

IrrSetActiveCamera(CameraObject)

IrrSetCameraFOV

Syntax

IrrSetCameraFOV (camera as irr_camera, fov as single)

Description

Sets the field of vision of the camera a wide field of vision will give a distorted perspective, if the angle is too narrow the display will feel restricted. The value is in radians and has a default value of PI / 2.5

Example

IrrSetCameraFOV(CameraObject, PI / 2)

IrrSetCameraAspectRatio

Syntax

IrrSetCameraAspectRatio (camera as irr_camera, aspectRatio as single)

Description

Sets the aspect ratio of the camera in the same way you think of standard screens and widescreens. A widescreen usually has an aspect ratio of 16:9 or 16/9 = 1.78. The camera apect ratio is set up automatically however if you are using split screen effects you may need to change the camera aspect ratio.

Example

Lighting

Calls to create and effect lighting in the scene.

IrrAddLight		
Syntax irr_node = IrrAddLight (x as single, y as singlez as single, red as single, green as single, blue as single, size as single)		
Description Adds a light into scene to naturally illuminate your scene.		
X, Y and Y defines the coordinates of the light in the scene.		
Red, Green and Blue define the intensities of the lighting for those colors. This is a fractional number ranging from 0 upwards the higher the value the brighter the light.		
Size specifies the radius of effect of the light		
Example		
WarningLight = IrrAddLight (0, 100, 50, 0.5,0.5,0.5, 50)		

IrrSetAmbientLight

Syntax

IrrSetAmbientLight (Red as single, Green as single, Blue as single)

Description

Sets the ambient lighting level in the scene, ambient light casts light evenly across the entire scene and can be used to increase the overall lighting level. If should never be greater that the brightness of the darkest area of your scene, it can however reduce the number of lights you need in the scene.

The Red, Green and Blue components of this lighting is supplied as integers in

the range or 0 to 255

Example

IrrSetAmbientLight(72, 64, 64)

IrrSetLightAmbientColor

Syntax

IrrSetLightAmbientColor(Light as irr_node, Red as single, Green as single, Blue as single)

Description

Ambient color emitted by the light, ambient light casts light evenly across the entire scene and can be used to increase the overall lighting level. If should never be greater that the brightness of the darkest area of your scene, it can however reduce the number of lights you need in the scene.

The Red, Green and Blue components of this lighting is supplied as singles specifying the brightness in each color channel

Example

IrrSetLightAmbientColor(SceneLight, 1.0, 0.1, 0.7)

IrrSetLightAttenuation

Syntax

IrrSetLightAttenuation(Light as irr_node, Red as single, Green as single, Blue as single)

Description

Changes the light strength fading over distance. Good values for distance effects use (1.0, 0.0, 0.0) and simply add small values to the second and third element.

Example

IrrSetLightAttenuation(SceneLight, 1.0, 0.08, 0.07)

IrrSetLightCastShadows

Syntax

IrrSetLightCastShadows(Light as irr_node, cast_shadows as uinteger)

Description

Specifies whether the light casts shadows in the scene or not. Shadowing must be enabled in the IrrStart call and also on the nodes in the scene.

Example

IrrSetLightCastShadows(SceneLight, IRR_ON)

IrrSetLightDiffuseColor

Syntax

IrrSetLightDiffuseColor(Light as irr_node, Red as single, Green as single, Blue as single)

Description IrrSetLightDiffuseColor

The Red, Green and Blue components of this lighting is supplied as singles specifying the brightness in each color channel

Example

IrrSetLightDiffuseColor(SceneLight, 1.0, 1.0, 0.8)

IrrSetLightFalloff

Syntax

IrrSetLightFalloff(Light as irr_node, Falloff as single)

Description

The light strength's decrease between Outer and Inner cone.

Example

IrrSetLightFalloff(SceneLight, 0.8)

IrrSetLightInnerCone

Syntax

IrrSetLightInnerCone(Light as irr_node, InnerCone as single)

Description

The angle of the spot's inner cone. Ignored for other lights.

Example

IrrSetLightInnerCone(SceneLight, 0.4)

IrrSetLightOuterCone

Syntax

IrrSetLightOuterCone(Light as irr_node, OuterCone as single)

Description

The angle of the spot's outer cone. Ignored for other lights.

Example

IrrSetLightOuterCone(SceneLight, 0.9)

IrrSetLightRadius

Syntax

IrrSetLightRadius(Light as irr_node, Radius as single)

Description

Radius of light. Everything within this radius be be lighted. If some artefacts can be seen when the radius is changed in this instance simply make the radius a little large

Example

IrrSetLightRadius(SceneLight, 50.2)

IrrSetLightSpecularColor

Syntax

IrrSetLightSpecularColor(Light as irr_node, Red as single, Green as single,

Blue as single)

Description

Sets the ambient lighting level in the scene, ambient light casts light evenly across the entire scene and can be used to increase the overall lighting level. If should never be greater that the brightness of the darkest area of your scene, it can however reduce the number of lights you need in the scene.

The Red, Green and Blue components of this lighting is supplied as singles specifying the brightness in each color channel

Example

IrrSetLightSpecularColor(SceneLight, 1.0, 1.0, 1.0)

IrrSetLightType

Syntax

IrrSetLightType(Light as irr_node, Light_type as E_LIGHT_TYPE)

Description

The type of the light. All lights default to a point light but can be changed with this setting to one of the following values: -

ELT_POINT ELT_SPOT ELT_DIRECTIONAL

Example

IrrSetLightType(SceneLight, ELT_SPOT)

Terrain

Calls to create and alter the properties of terrain meshes, special nodes that are used to create large expansive landscapes.

IrrAddTerrain

Syntax

irr_terrain = IrrAddTerrain (path as zstring ptr, xPosition as single = 0.0, yPosition as single = 0.0, zPosition as single = 0.0, xRotation as single = 0.0, zRotation as single = 0.0, xScale as single = 1.0, yScale as single = 1.0, zScale as single = 1.0, vertexAlpha as integer = 255, vertexRed as integer = 255, vertexGreen as integer = 255, vertexBlue as integer = 255, smoothing as integer = 0, maxLOD as integer = 5, patchSize as IRR_TERRAIN_PATCH_SIZE = ETPS_17)

Description

Creates a terrain object from a gray scale bitmap where bright pixels are high points on the terrain and black pixels are low points. You will inevitablly have to rescale the terrain during the call or after it is created. The Terrain object is a special dynamic mesh whose resoloution is reduced in the distance to reduce the number of triangles it consumes.

Path is the filename of a gray scale image used to define the contours of the surface.

xPosition, yPosition and zPosition define the position of the terrain xRotation, yRotation and zRotation define the rotation of the terrain

xScale, xScale and xScale define the scale of the terrain

vertexAlpha, vertexRed, vertexGreen, vertexBlue, define the vertex color of all points in the terrain.

smoothing allows you to define whether the contours of the surface of the terrain are smoothed over.

maxLOD and patchsize control the properties of the level of detail calculations applied to the terrain, it is recommended that these are left at default values.

Example

TerrainNode = IrrAddTerrain("CanyonsHeightField.bmp"

IrrAddTerrainTile

Syntax

irr_terrain = IrrAddTerrainTile (image as irr_image, tileSize as integer = 256, dataX as integer = 0, dataY as integer = 0, xPosition as single = 0.0, yPosition as single = 0.0, zPosition as single = 0.0, xRotation as single = 0.0, yRotation as single = 0.0, zRotation as single = 0.0, xScale as single = 1.0, yScale as single = 1.0, zScale as single = 1.0, smoothing as integer = 1, maxLOD as integer = 5, patchSize as IRR_TERRAIN_PATCH_SIZE = ETPS_17)

Description

Creates a tilable terrain object from a gray scale bitmap where bright pixels are high points on the terrain and black pixels are low points. You will inevitablly have to rescale the terrain during the call or after it is created. The Terrain object is a special dynamic mesh whose resoloution is reduced in the distance to reduce the number of triangles it consumes.

Unlike the origonal terrain object the tileable terrain object can be attached to other terrain tile objects without being affected by cracks between tiles caused by the level of detail mechanism. When working with tile terrains it should be noted that the terrain is internally divided up into patches that are patchSize - 1 and there is always one invisible row of patches at the top and left of the terrain. Essentially this means that if your tileSize is 128 x 128 the visible size of your terrain will be 112 x 112 (with a patchSize of ETPS_17)

Note: Tiled Terrain object can be automatically control with the Zone Manager objects please refer to them for further details.

Image is an image file loaded with IrrGetImage and containing a gray scale image used to define the contours of the surface.

TileSize defines the size of the terrain independantly of the size of the image used to create it

xPosition, yPosition and zPosition define the position of the terrain xRotation, yRotation and zRotation define the rotation of the terrain

xScale, xScale and xScale define the scale of the terrain smoothing allows you to define whether the contours of the surface of the terrain are smoothed over.

maxLOD and patchsize control the properties of the level of detail calculations applied to the terrain, it is recommended that these are left at default values.

Example

IrrAddSphericalTerrain

Syntax

irr_terrain = IrrAddSphericalTerrain (topPath as zstring ptr, frontPath as zstring ptr, backPath as zstring ptr, leftPath as zstring ptr, rightPath as zstring ptr, bottomPath as zstring ptr, xPosition as single = 0.0, yPosition as single = 0.0, zPosition as single = 0.0, xRotation as single = 0.0, yRotation as single = 0.0, zRotation as single = 0.0, xScale as single = 1.0, yScale as single = 1.0, zScale as single = 1.0, vertexAlpha as integer = 255, vertexRed as integer = 255, vertexGreen as integer = 255, vertexBlue as integer = 255, smoothing as integer = 0, spherical as integer = 0, maxLOD as integer = 5, patchSize as IRR_TERRAIN_PATCH_SIZE = ETPS_17)

Description

Creates a spherical terrain that represents a planetary body. When using this terrain it is better to think of it as a cube rather than a sphere, in fact it **is** a cube that is distorted so that its surface becomes spherical, like a cube it has a top, bottom, left, right, front and back and co-ordinates are thought of as being at position X,Y on cube face N. In someways this makes working with placing things on the object simpler as you can think of it as six flat surfaces.

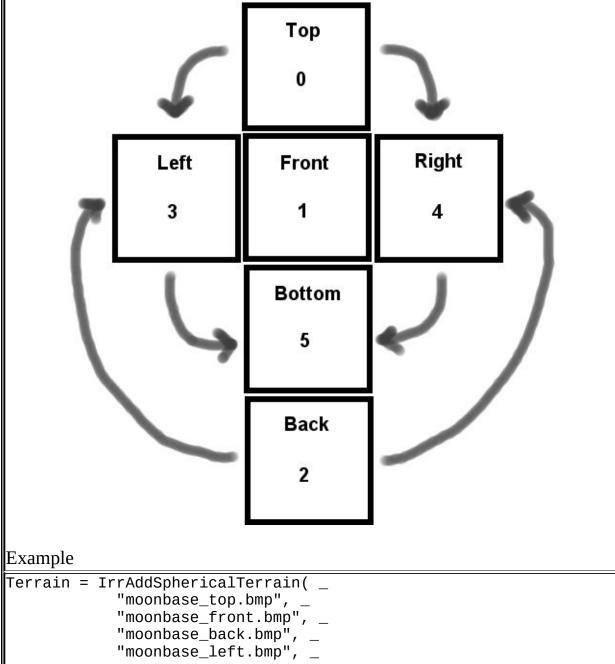
The first six paths are the path of six gray scale bitmaps where bright pixels are high points on the terrain and black pixels are low points.

The position, rotation and scale of the terrain are specified with the next series of parameters.

Four parameters are used to set the vertex color of all the verticies in the terrain.

Smoothing is used to smooth out the contours of the hills in the terrain. maxLOD and patchSize allow you to adjust the level of detail within the terrain although it is usually best to leave these to default values.

When creating heightmaps for the faces of the terrain you will need to ensure that the height of pixels at the edge of adjoining sides of the terrain are the same otherwise large visible cracks will appear at the edges of the faces, the easiest way to do this is to create terrain texture and then copy and/or rotate it onto its adjacent face. You can get some suprisingly effective planets and asteroids with textures as small as 32x32 but the object also runs well with a terrain size at the maximum 256 x 256.



"moonbase_right.bmp", _
"moonbase_bottom.bmp", _
px,py,pz, rx,ry,rz, 64.0,64.0,64.0, _
0, 255, 255, 255, -30, 0, 4, ETPS_17)

IrrGetTerrainHeight

Syntax

single = IrrGetTerrainHeight (terrain as irr_terrain, X as single, Y as single)

Description

Get the height of a point on a terrain. This can be a particularly fast and accurate way to move an object over a terrain.

Example

Y = IrrGetTerrainHeight (TerrainNode, X, Z)

IrrScaleTexture

Syntax

IrrScaleTexture (terrain as irr_terrain, X as single, Y as single)

Description

As a terrain object is a particularly huge mesh when textured are applied to it they look extremely pixelated. To get over this effect a terrain object can have two materials applied to it, one to give general surface color and a second that is copied across the surface like tiles to give a rough detailed texture. This call specifies the scaling of this detail texture.

Example

IrrScaleTexture (TerrainNode, 20, 20)

IrrGetTerrainTileHeight

Syntax

single = IrrGetTerrainTileHeight (terrain as irr_terrain, X as single, Y as single)

Description

Get the height of a point on a terrain tile. This can be a particularly fast and accurate way to move an object over a terrain.

Example

Y = IrrGetTerrainTileHeight (TerrainNode, X, Z)

IrrScaleTileTexture

Syntax

IrrScaleTileTexture (terrain as irr_terrain, X as single, Y as single)

Description

As a tile terrain object is a particularly huge mesh when textured are applied to it they look extremely pixelated. To get over this effect a terrain object can have two materials applied to it, one to give general surface color and a second that is copied across the surface like tiles to give a rough detailed texture. This call specifies the scaling of this detail texture.

Example

IrrScaleTileTexture (TerrainNode, 20, 20)

IrrAttachTile

Syntax

IrrAttachTile (terrain as irr_terrain, neighbouring_terrain as irr_terrain, edge as integer)

Description

Set the adjacent tile to this tile node. To avoid cracks appearing between tiles, tiles need to know which tiles are their neighbours and which edges they are attached too.

Example

```
IrrAttachTile( TerrainNorth, TerrainSouth, TOP_EDGE )
IrrAttachTile( TerrainSouth, TerrainNorth, BOTTOM_EDGE )
```

IrrSetTileStructure

Syntax

IrrSetTileStructure (terrain as irr_terrain, image as irr_image, x as integer, y as integer)

Description

Loads the tile structure from the supplied image file. Unlike the image in the origonal call to create a terrain tile this image has a different structure. The image should be in RGBA format, the alpha value is used to set the height of the terrain and the RGB values are used to set the color of the verticies. This can either be for loading precalculated lighting into the scene or it can be used with the new IRR_EMT_FOUR_DETAIL_MAP material type to define the weight of each of the greyscale detail maps in the RGB channels of the detail map. The x and y values can be used to load the structure from a specific point on the bitmap.

Example

IrrSetTileStructure(TerrainCove, CoveStructure, 0, 0)

IrrSetTileColor

Syntax

IrrSetTileColor(terrain as irr_terrain, image as irr_image, x as integer, y as integer)

Description

Loads the tile vertex colors from the supplied image file. The RGB values are used to set the color of the verticies. This can either be for loading precalculated lighting into the scene or it can be used with the new IRR_EMT_FOUR_DETAIL_MAP material type to define the weight of each of the greyscale detail maps in the RGB channels of the detail map. The x and y values can be used to load the structure from a specific point on the bitmap.

Example

IrrSetTileColor(TerrainCove, CoveStructure, 0, 0)

IrrScaleSphericalTexture

Syntax

IrrScaleSphericalTexture (terrain as irr_terrain, X as single, Y as single)

Description

As the surfaces of a sphereical terrain object are a particularly huge mesh when textures are applied to them they look extremely pixelated. To get over this effect a spherical terrain object can have two materials applied to it, one to give general surface color and a second that is copied across the surface like tiles to give a rough detailed texture. This call specifies the scaling of this detail texture.

Example

IrrScaleSphericalTexture (SphericalTerrainNode, 20, 20)

IrrSetSphericalTerrainTexture

Syntax

IrrSetSphericalTerrainTexture (terrain as irr_terrain, topTexture as irr_texture, frontTexture as irr_texture, backTexture as irr_texture, leftTexture as irr_texture, rightTexture as irr_texture, bottomTexture as irr_texture, materialIndex as uinteger)

Description

Apply six textures to the surface of a spherical terrain. By using the material index you can set the color or the detail maps

Example

IrrSetSphericalTerrainTexture (<pre>TerrainNode, "moobbase_col_top.bmp", "moobbase_col_front.bmp", "moobbase_col_back.bmp", "moobbase_col_left.bmp",</pre>
	<pre>"moobbase_col_right.bmp", _ "moobbase_col_bottom.bmp", _ 0)</pre>

IrrLoadSphericalTerrainVertexColor

Syntax

IrrLoadSphericalTerrainVertexColor (terrain as irr_terrain, topMap as irr_image, frontMap as irr_image, backMap as irr_image, leftMap as irr_image, rightMap as irr_image, bottomMap as irr_image)

Description

Apply six images to the vertex colors of the faces, this is useful for setting the verticies so that they can be used with simple terrain spattering described in the section on tiled terrains above.

Example

-
IrrLoadSphericalTerrainVertexColor (TerrainNode, _
"moobbase_vert_top.bmp", _
"moobbase_vert_front.bmp", _
"moobbase_vert_back.bmp", _
"moobbase_vert_left.bmp", _
"moobbase_vert_right.bmp", _
"moobbase_vert_bottom.bmp")

IrrGetSphericalTerrainSurfacePosition

Syntax

IrrGetSphericalTerrainSurfacePosition (terrain as irr_terrain, face as integer, logicalX as single, logicalZ as single, X as single, Y as single, Z as single)

Description

Get the surface position of a logical point on the terrain. You supply a face number and a logical X, Y position on that face and this call will return the height of that point on the terrain sphere inside the X, Y, Z parameters.

Note: By subtracting the center of the sphere from this co-ordinate and converting this vector to angles you can find the upward direction of the point on the surface.

Example

```
IrrGetSphericalTerrainSurfacePosition ( TerrainNode, IRR_TOP_FACE,
buggyX, buggyZ, X, Y, Z )
```

IrrGetSphericalTerrainSurfacePositionAndAngle

Syntax

IrrGetSphericalTerrainSurfacePosition (terrain as irr_terrain, face as integer, logicalX as single, logicalZ as single, X as single, Y as single, Z as single,

RotationX as single, RotationY as single, RotationZ as single)

Description

Get the surface position and angle of a logical point on the terrain. This is not the normal of the surface but essentially the angles to the gravitational center.

Example

```
IrrGetSphericalTerrainSurfacePositionAndAngle ( Terrain, F, I, J,
PX,PY,PZ, RX,RY,RZ )
```

IrrGetSphericalTerrainLogicalSurfacePosition

Syntax

IrrGetSphericalTerrainSurfacePosition (terrain as irr_terrain, face as integer, logicalX as single, logicalZ as single, X as single, Y as single, Z as single)

Description

Convert a co-ordinate into a logical Spherical terrain position. Thanks for the example from "David" posted on Infinity-Universe forum

Please note that this calculation is not 100% accurate, it is advised that the translation is done at altitude and the difference either ignored or blended as the observer decends.

Note: The height above the surface can be calculated simply by calculating the length of the center of the planet to the surface and then the center of the planet to the space coordinate and subracting the two

Note: The momentum could be calculated by converting two samples and then measing the difference in height and X and Z on the face

Example

```
IrrGetSphericalTerrainLogicalSurfacePosition ( Terrain, X, Y, Z,
face, LX, LZ )
```

Particles

Calls to control the appearance and follow of particles in particle systems.

IrrSetMinParticleSize

Syntax

IrrSetMinParticleSize (particle emitter as irr_particle_emitter, X as single, Y as single)

Description

Particles in a particle system are simple 2 dimensional billboard like objects, this sets the size of these particles, larger particles can be effective and use less resources however they can look blocky if taken too far.

Example

IrrSetMinParticleSize (SmokeEmitter, 5, 5)

IrrSetMaxParticleSize

Syntax

IrrSetMaxParticleSize (particle emitter as irr_particle_emitter, X as single, Y as single)

Description

Particles in a particle system are simple 2 dimensional billboard like objects, this sets the size of these particles, larger particles can be effective and use less resources however they can look blocky if taken too far.

Example

```
IrrSetMaxParticleSize ( SmokeEmitter, 15, 15 )
```

IrrAddParticleEmitter

Syntax irr_emitter = IrrAddParticleEmitter (particle system as irr_particle_system, settings as IRR_PARTICLE_EMITTER)

Description

Adds a particle emitter to the particle system, this creates particles and controls how they move and when they are to be removed. It requires a very large number of parameters to define this flexible effect and as such these parameters are stores in a special IRR_PARTICLE_EMITTER structure.

Example

MyEmitter = IrrAddParticleEmitter(SmokeParticles, SmokeEmitter

IrrAddAnimatedMeshSceneNodeEmitter

Syntax

irr_emitter = IrrAddAnimatedMeshSceneNodeEmitter(particle_system as irr_particle_system, node as irr_node, use_normal_direction as uinteger, normal_direction_modifier as single, emit_from_every_vertex as integer, settings as IRR_PARTICLE_EMITTER)

Description

Creates a particle emitter for an animated mesh scene node

Parameters:

node - Pointer to the animated mesh scene node to emit particles from useNormalDirection - If true, the direction of each particle created will be the normal of the vertex that it's emitting from. The normal is divided by the normalDirectionModifier parameter, which defaults to 100.0f. normalDirectionModifier - If the emitter is using the normal direction then the normal of the vertex that is being emitted from is divided by this number. everyMeshVertex - If true, the emitter will emit between min/max particles every second, for every vertex in the mesh, if false, it will emit between min/max particles from random vertices in the mesh.

A large number of additional parameters are also required to define this flexible effect and as such these parameters are stores in a special IRR_PARTICLE_EMITTER structure. The box size properties of this structure

are unused in this call

Example

```
MyEmitter = IrrAddAnimatedMeshSceneNodeEmitter ( SmokeParticles,
SceneNode, 1, 0.25, 0, SmokeEmitter )
```

IrrAddFadeOutParticleAffector

Syntax

irr_affector = IrrAddFadeOutParticleAffector (particle_system as irr_particle_system)

Description

Adds a fade out affector to the particle system, this fades the particles out as they come to the end of their lifespan and stops them 'popping' out of existance. This creates a convincing effect for fire and smoke in particular.

Example

IrrAddGravityParticleAffector

Syntax

irr_affector = IrrAddGravityParticleAffector (particle system as irr_particle_system, x as single, y as single, z as single)

Description

Adds a gravity affector to the particle system, this gradually pulls the particles in the direction of the effect, although it is called a gravity effector it can be used to make a wind effect and have the particles drift off to the side.

X, Y and Z define the force that is applied to the particles over time.

Example

MyAffector = IrrAddGravityParticleAffector(SmokeParticles, -0.1, 0, 0)

IrrAddParticleAttractionAffector

Syntax

irr_affector = IrrAddParticleAttractionAffector(particle_system as irr_particle_system, x as Single, y as Single, z as Single, speed as Single = 1.0, attract as uinteger = 1, affectX as uinteger = 1, affectY as uinteger = 1, affectZ as uinteger = 1)

Description

Creates a point attraction affector. This affector modifies the positions of the particles and attracts them to a specified point at a specified speed per second.

Parameters:

x,y,z - Point to attract particles to.

speed - Speed in units per second, to attract to the specified point.

attract - Whether the particles attract or detract from this point use the constants IRR_ATTRACT or IRR_REPEL (defaults to IRR_ATTRACT)

affectX - Whether or not this will affect the X position of the particle, use 1 to effect the position and 0 to leave it unaffected (defaults to true).

affectY - Whether or not this will affect the Y position of the particle, use 1 to effect the position and 0 to leave it unaffected (defaults to true).

affectZ - Whether or not this will affect the Z position of the particle, use 1 to effect the position and 0 to leave it unaffected (defaults to true).

Example

MyAffector = IrrAddParticleAttractionAffector(SmokeParticles, 0.0, 10.0, 0.0, 20.0, IRR_ATTRACT, 1, 1, 1)

IrrCreateRotationAffector

Syntax

irr_affector = IrrCreateRotationAffector (particle_system as irr_particle_system, Speed_X as Single, Speed_Y as Single, Speed_Z as Single, pivot_X as Single, pivot_Y as Single, pivot_Z as Single)

Description

Creates a rotation affector. This affector modifies the positions of the particles and attracts them to a specified point at a specified speed per second.

Parameters:

speed x,y,z - Rotation in degrees per second

pivot x,y,z - Point to rotate the particles around

Example

```
MyAffector = IrrCreateRotationAffector( SmokeParticles, -120.0,
0.0, 0.0, 0.0, 0.0, 0.0 )
```

IrrAddStopParticleAffector

Syntax

irr_affector = IrrAddStopParticleAffector (particle_system as irr_particle_system, time as uinteger, emitter as irr_emitter)

Description

The stop particle affector waits for the specified period of time to elapse and then stops the specified emitter emitting particles by setting its minimum and maximum particle emission rate to zero. The emitter can easily be started up again by changing its emission rate.

Parameters:

Time - The number of milliseconds to elapse before the particles are stopped Emitter - The particle generating object to stop

Example

```
MyAffector = IrrAddStopParticleAffector(    SmokeSystem, 1000,
smoke_emitter )
```

IrrAddParticlePushAffector

Syntax

irr_affector = IrrAddParticlePushAffector (particle_system as irr_particle_system, x as single, y as single, z as single, speedX as single, speedY as single, speedZ as single, far as single, near as single, column as single, distant as integer)

Description

Creates a point push affector. This affector modifies the positions of the particles and pushes them toward or away from a specified point at a specified speed per second. The strength of this effect is adjusted by a near and a far distance. Beyond the far distance the particle is not effected at all and the closer

you get to the center of the effect the stronger the force is.

If a near distance is defined (a value greater than 0.0) the effect is somewhat different, particles closer to the center than the near distance are not effected at all, and the stongest point of the effect is always halfway between the near and far limits, for example if your near distance was 25.0 and your far distance was 75.0 the strongest force would be applied to particles at a distance of 50.0

If a column width is defined the effect will only take place in a vertical column that is that wide, this is useful for fountains of water where as the water spreads out of the column a gravity affector can take over.

By adjusting these parameters and the strength you can create columns, spheres, shells and rings of effect that can, in combination, push particles in complex motions

Parameters:

x, y, z - Point to attract particles to or repel particles away from speedX, speedY, speedZ - A vector describing the strength of the effect Far - Furthest distance of effect

Near - Closest distance of effect

Column - The width of a vertical column in which the push affector has influence, somewhat like the column of water in a fountain Distant - Use IRR_ON to apply the same force in the same directionto all

particles and use IRR_OFF to apply a force that radiates away from the center of the effect

Example

```
MyAffector = IrrAddParticlePushAffector ( ColumnOfSmoke, 0, 0, 0, 0, 100, 0, 100, 0.0, 0.0, IRR_OFF )
```

IrrAddColorMorphAffector

Syntax

irr_affector = IrrAddColorMorphAffector (particle_system as irr_particle_system, numberOfParticles as uinteger, particlecolors as uinteger ptr, particletimes as uinteger ptr, smooth as uinteger)

Description This clever effect by Dark Kilauea that allows you to provide an array of colors and an optional array of times that effect the color of the particle over its lifetime, the particle could start off bright orange and fade away into grey and then black for example.		
Parameters: numEntries - the number of entries in the supplied table colors - the table of colors time - the table of times at which each color becomes active smooth - Use IRR_ON to smoothly blend between colors and use IRR_OFF to sharply switch between colors		
Example		
DIM colors(0 to 4) as uinteger = { ' yellow white	IrrMakeARGB(0,255,255,128), _	
yellow	IrrMakeARGB(0,255,128,0), _ '	
orange	IrrMakeARGB(0,128,64,0), _ '	
slight blue	IrrMakeARGB(0,0,0,128), _ '	
	<pre>IrrMakeARGB(255,0,0,0) } '</pre>	
black and faded DIM times(0 to 4) as uinteger = {500, 800, 1250, 1500, 2000 }		
MyAffector = IrrAddColorMorphAffector(Fire.particles, 5, @colors(0), @times(0), IRR_ON)		

IrrAddSplineAffector

Syntax

irr_affector = IrrAddSplineAffector (particle_system as irr_particle_system, VertexCount as uinteger, verticies as IRR_VERT ptr, speed as single, tightness as single, attraction as single, deleteAtEnd as uinteger)

Description

This clever effect by Dark Kilauea that allows you to create an affector that moves the particles along the path of a spline for very controled and complex particle motion. Parameters:

VertexCount - Is the number of points in your spline

Verticies - Is an array of IRR_VERT objects defining the X,Y and Z position of points

Speed - is the speed with which particles are moved along the spline tightness - is the tightness of the curve of the spline

attraction - is how closely the particles are atracted to the curve of the spline deleteAtEnd - Use IRR_ON to delete the particles when they reach the end of the spline and use IRR_OFF to allow the particles to be deleted naturally.

Example

```
DIM splineverticies(0 to 3) as IRR_VERT
splineverticies(0).x = 0.0 : splineverticies(0).y = 0.0 :
splineverticies(0).z = 0.0
splineverticies(1).x = 0.0 : splineverticies(1).y = 20.0 :
splineverticies(1).z = 25.0
splineverticies(2).x = 0.0 : splineverticies(2).y = 40.0 :
splineverticies(2).z = -25.0
splineverticies(3).x = 0.0 : splineverticies(3).y = 60.0 :
splineverticies(3).z = 0.0
IrrAddSplineAffector ( NeonLight.particles, 4,
@splineverticies(0), 2.0, 1.0, 5.0, IRR_ON )
```

IrrRemoveAffectors

Syntax

IrrRemoveAffectors (particle system as irr_particle_system)

Description

Removes all affectors from a particle system, you might use this if you wanted to change the direction or strength of the wind for example.

Example

IrrRemoveAffectors(SmokeParticles)

IrrSetParticleEmitterDirection

Syntax

IrrSetParticleEmitterDirection(particle_emitter as irr_emitter, x as single, y as

single, z as single)

Description Set direction the emitter emits particles.

Example

IrrSetParticleEmitterDirection(MyEmitter, 0.0, 0.4, 0.0)

IrrSetParticleEmitterMinParticlesPerSecond

Syntax

IrrSetParticleEmitterMinParticlesPerSecond(particle_emitter as irr_emitter, min_particles_per_second as uinteger)

Description

Set minimum number of particles the emitter emits per second.

Example

IrrSetParticleEmitterMinParticlesPerSecond(MyEmitter, 32)

IrrSetParticleEmitterMaxParticlesPerSecond

Syntax

IrrSetParticleEmitterMaxParticlesPerSecond(particle_emitter as irr_emitter, max_particles_per_second as uinteger)

Description

Set maximum number of particles the emitter emits per second.

Example

IrrSetParticleEmitterMaxParticlesPerSecond(MyEmitter, 100)

IrrSetParticleEmitterMinStartColor

Syntax

IrrSetParticleEmitterMinStartColor(particle_emitter as irr_emitter, Red as uinteger, Green as uinteger, Blue as uinteger)

Description Set minimum starting color for particles.

Example

IrrSetParticleEmitterMinStartColor(MyEmitter, 255, 192, 128)

IrrSetParticleEmitterMaxStartColor

Syntax

IrrSetParticleEmitterMaxStartColor(particle_emitter as irr_emitter, Red as uinteger, Green as uinteger, Blue as uinteger)

Description

Set maximum starting color for particles.

Example

IrrSetParticleEmitterMaxStartColor(MyEmitter, 255, 192, 128)

IrrSetParticleAffectorEnable

Syntax

IrrSetParticleAffectorEnable(particle_affector as irr_affector, Enable as uinteger)

Description

Enable or disable an affector. Setting the value to 1 enables the affector, setting it to 0 disables it. IRR_ON and IRR_OFF can be used also.

Example

IrrSetParticleAffectorEnable(MyAffector, IRR_OFF)

IrrSetFadeOutParticleAffectorTime

Syntax

IrrSetFadeOutParticleAffectorTime(particle_affector as irr_affector, FadeFactor as float)

Description

Alter the fadeout affector changing the fade out time.

Example

IrrSetFadeOutParticleAffectorTime(MyAffector, 2000.0)

IrrSetFadeOutParticleAffectorTargetColor

Syntax

IrrSetFadeOutParticleAffectorTargetColor(particle_affector as irr_affector, Red as uinteger, Green as uinteger, Blue as uinteger)

Description

Alter the fadeout affector changing the target color to the affector fades to over time.

Example

IrrSetFadeOutParticleAffectorTargetColor(MyAffector, 16, 8, 0)

IrrSetGravityParticleAffectorDirection

Syntax

IrrSetGravityParticleAffectorDirection(particle_affector as irr_affector, x as single, y as single, z as single)

Description

Alter the direction and force of gravity for a gravity affector.

Example

IrrSetGravityParticleAffectorDirection(MyAffector, 0.2, 0.1, 0.0

)

IrrSetGravityParticleAffectorTimeForceLost

Syntax

IrrSetGravityParticleAffectorTimeForceLost(particle_affector as irr_affector, time_force_lost as single)

Description

Set the time in milliseconds when the gravity force is totally lost and the particle does not move any more.

Example

IrrSetGravityParticleAffectorTimeForceLost(MyAffector, 800.0)

IrrSetParticleAttractionAffectorAffectX

Syntax

IrrSetParticleAttractionAffectorAffectX(particle_affector as irr_affector, affect as uinteger)

Description

Set whether or not an atraction affector will affect particles in the X direction.. Setting the value to 1 enables the effect, setting it to 0 disables it. IRR_ON and IRR_OFF can be used also.

Example

IrrSetParticleAttractionAffectorAffectX(MyAffector, IRR_ON)

IrrSetParticleAttractionAffectorAffectY

Syntax

IrrSetParticleAttractionAffectorAffectY(particle_affector as irr_affector, affect as uinteger)

Description

Set whether or not an atraction affector will affect particles in the Y direction.. Setting the value to 1 enables the effect, setting it to 0 disables it. IRR_ON and IRR_OFF can be used also.

Example

IrrSetParticleAttractionAffectorAffectY(MyAffector, IRR_ON)

IrrSetParticleAttractionAffectorAffectZ

Syntax

IrrSetParticleAttractionAffectorAffectZ(particle_affector as irr_affector, affect

as uinteger)

Description

Set whether or not an atraction affector will affect particles in the Z direction.. Setting the value to 1 enables the effect, setting it to 0 disables it. IRR_ON and IRR_OFF can be used also.

Example

IrrSetParticleAttractionAffectorAffectZ(MyAffector, IRR_ON)

IrrSetParticleAttractionAffectorAttract

Syntax

IrrSetParticleAttractionAffectorAttract(particle_affector as irr_affector, affect as uinteger)

Description

Set whether or not the particles are attracted or repelled from an attractor effector.. Use the values IRR_ATTRACT and IRR_REPEL for convienience.

Example

IrrSetParticleAttractionAffectorAttract(MyAffector, IRR_ATTRACT)

IrrSetParticleAttractionAffectorPoint

Syntax

IrrSetParticleAttractionAffectorPoint(particle_affector as irr_affector, x as single, y as single, z as single)

Description

Set the point that particles will attract to when affected by this attractor affector.

Example

IrrSetParticleAttractionAffectorPoint(MyAffector, IRR_ATTRACT)

IrrSetRotationAffectorPivotPoint

Syntax

IrrSetRotationAffectorPivotPoint(particle_affector as irr_affector, x as single, y as single, z as single)

Description

Set the point that particles will rotate about when affected by this rotation affector.

Example

IrrSetRotationAffectorPivotPoint(MyAffector, IRR_ATTRACT)

IrrSetFurthestDistanceOfEffect

Syntax

IrrSetFurthestDistanceOfEffect(particle_affector as irr_affector, newDistance as single)

Description Set the furthest distance of effect on particles affected by the push affector.

Example

IrrSetFurthestDistanceOfEffect(MyAffector, 100.0)

IrrSetNearestDistanceOfEffect

Syntax

IrrSetNearestDistanceOfEffect(particle_affector as irr_affector, newDistance as single)

Description

Set the nearest distance of effect on particles affected by the push affector.

Example

IrrSetNearestDistanceOfEffect(MyAffector, 10.0)

IrrSetColumnDistanceOfEffect

Syntax

IrrSetColumnDistanceOfEffect(particle_affector as irr_affector, newDistance

as single)

Description

Set the column distance of effect on particles affected by the push affector.

Example

IrrSetColumnDistanceOfEffect(MyAffector, 20.0)

IrrSetCenterOfEffect

Syntax

IrrSetCenterOfEffect(particle_affector as irr_affector, x as single, y as single, z as single)

Description

Set the center of the effect of particles affected by the push affector.

Example

IrrSetCenterOfEffect(MyAffector, 0.0, PushHeight, 0.0)

IrrSetStrengthOfEffect

Syntax

IrrSetStrengthOfEffect(particle_affector as irr_affector, x as single, y as single, z as single)

Description

Set the strength of the effect of particles affected by the push affector.

Example

IrrSetStrengthOfEffect(MyAffector, PipeVelocity, 0.0, 0.0)

Irrlicht Graphical User Interface

Calls to add graphical user interface objects to the screen that can be drawn with a single call. At the moment this section is awaiting further development in the wrapper.

IrrGUIClear
Syntax IrrGUIClear ()
Description Clears all GUI objects from the display.
Example
IrrGUIClear()

IrrGUIRemove

Syntax

IrrGUIRemove (object as IRR_GUI_OBJECT)

Description Removes the specified GUI object from the display.

Example

IrrGUIRemove(myButton)

IrrGUIGetText

Syntax IrrGUIGetText (object as IRR_GUI_OBJECT)

Description Gets the text associated with a GUI object.. Example

DIM myString as wstring = IrrGUIGetText(myEditBox)

IrrGUISetText

Syntax

IrrGUISetText (object as IRR_GUI_OBJECT, text as wstring)

Description Sets the text of a GUI object..

Example

DIM fpsString as wstring * 256 fpsString = "FPS: " + Str(IrrGetFPS) IrrGUISetText(myButton, fpsString)

IrrAddWindow

Syntax

IrrAddWindow (title as wstring ptr, Top X as integer, Top Y as integer, Bottom X as integer, Bottom Y as integer, modal as uinteger, parent as IRR_GUI_OBJECT) as IRR_GUI_OBJECT

Description

Creates an empty window that can form the frame to contain other controls.

Title is a wide string that contains the title of the window.

Top X, Top Y, Bottom X and Bottom Y define a box in which the window is drawn

Modal determines if the window locks out the rest of the interface until it is closed: -

IRR_GUI_MODAL IRR_GUI_NOT_MODAL

Parent defines the parent object of this window. This can be ommited if the window has no parent.

Example

windowObject = IrrAddWindow("A Window", 4,0,200,64, IRR_GUI_MODAL
)

IrrAddStaticText

Syntax

IrrAddStaticText (text as wstring ptr, Top X as integer, Top Y as integer, Bottom X as integer, Bottom Y as integer, border as uinteger, wordwrap as uinteger, parent as IRR_GUI_OBJECT) as IRR_GUI_OBJECT

Description

Creates a static text object on the Graphical User Interface, this simply displays the specifed text in the specified box.

Text is a wide string that contains the text you want to display.

Top X, Top Y, Bottom X and Bottom Y define a box in which the text is drawn

Border is used to draw a visible box around the text, its value should be either of: -

IRR_GUI_NO_BORDER IRR_GUI_BORDER

Word wrap is used to define whether text is to be wrapped around into a second line when it fills the width of the text box, its value should be either of: - IRR_GUI_NO_WRAP 0 IRR_GUI_WRAP 1

Parent defines the parent object of this window. This can be ommited if the object has no parent.

Example

```
statictextObject = IrrAddStaticText( "Hello World", 4,0,200,16,
NO_BORDER, NO_WRAP, windowObject )
```

1

IrrAddButton

Syntax

IrrAddButton (Top X as integer, Top Y as integer, Bottom X as integer, Bottom Y as integer, id as integer, text as wstring ptr, parent as IRR_GUI_OBJECT) as IRR_GUI_OBJECT

Description Add a clickable button object to the gui display.

Top X, Top Y, Bottom X and Bottom Y define a box in which the button is drawn

id specifies a unique numerical reference for the button so events can be identified as coming from this object

text specified the label assigned to the button

Parent defines the parent object of this window. This can be ommited if the object has no parent.

Example

buttonObject = IrrAddButton(16,16,96,32, 120, "My Button", windowObject)

IrrAddScrollBar

Syntax

IrrAddScrollBar (horizontal as integer, Top X as integer, Top Y as integer, Bottom X as integer, Bottom Y as integer, id as integer, currentValue as integer, maxValue as integer, parent as IRR_GUI_OBJECT) as IRR_GUI_OBJECT

Description Add a scrollbar object to the GUI display.

Horizontal defines if the scrollbar is horizontal or vertical, acceptable values for this field are: -

IRR_GUI_HORIZONTAL IRR_GUI_VERTICAL

Top X, Top Y, Bottom X and Bottom Y define a box in which the scrollbar is drawn

id specifies a unique numerical reference for the scrollbar so events can be identified as coming from this object

currentValue specified the current setting of the scrollbar

maxValue specifies the maximum setting of the scrollbar

Parent defines the parent object of this window. This can be ommited if the object has no parent.

Example

scrollbarObject = IrrAddScrollBar(IRR_GUI_HORIZONTAL, 16,16,96,32, 120, 128, 156, windowObject)

IrrAddListBox

Syntax

IrrAddListBox (horizontal as integer, Top X as integer, Top Y as integer, Bottom X as integer, Bottom Y as integer, id as integer, background as integer, parent as IRR_GUI_OBJECT) as IRR_GUI_OBJECT

Description Add a listbox object containing a list of items to the gui display.

horizontal specifies whether the scrollbar is oriented horizontally or vertically. acceptable values are: -

IRR_GUI_HORIZONTAL IRR_GUI_VERTICAL

Top X, Top Y, Bottom X and Bottom Y define a box in which the listbox is drawn

id specifies a unique numerical reference for the listbox so events can be identified as coming from this object

background specifies whether the background of the listbox should be drawn. acceptable values are: -

IRR_GUI_DRAW_BACKGROUND IRR_GUI_EMPTY_BACKGROUND

Parent defines the parent object of this window. This can be ommited if the object has no parent.

Example

listboxObject = IrrAddListBox(16,16,96,32, 120, IRR_GUI_DRAW_BACKGROUND, windowObject)

IrrAddListBoxItem

Syntax

IrrAddListBoxItem (listbox as IRR_GUI_OBJECT, text as wstring)

Description

Add a text element to a list box.

listbox defines the listbox gui object to add the string too.

text is the string containing the new item

Example

IrrAddListBoxItem(listboxObject, "Apples")

IrrInsertListBoxItem

Syntax

IrrInsertListBoxItem (parent as IRR_GUI_OBJECT, text as wstring, index as integer)

Description Insert a text element to a list box. listbox defines the listbox gui object to insert the string into.

text is the string containing the new item

index is the position at which to insert the item

Example

IrrInsertListBoxItem(listboxObject, "Pears", 3)

IrrRemoveListBoxItem

Syntax

IrrRemoveListBoxItem (parent as IRR_GUI_OBJECT, index as integer)

Description

Remove a text element from a list box.

listbox defines the listbox gui object to remove the string from.

index is the position of the item to be removed

Example

IrrRemoveListBoxItem(listboxObject, 2)

IrrSelectListBoxItem

Syntax

IrrSelectListBoxItem (parent as IRR_GUI_OBJECT, index as integer)

Description Select a text element in a list box.

listbox defines the listbox gui object to select the item within.

index is the position of the item to be removed

Example

IrrSelectListBoxItem(listboxObject, 1)

IrrAddEditBox

Syntax

IrrAddEditBox (text as wstring, horizontal as integer, Top X as integer, Top Y as integer, Bottom X as integer, Bottom Y as integer, id as integer, border as integer, password as integer, parent as IRR_GUI_OBJECT) as IRR_GUI_OBJECT

Description Add a editbox object containing a list of items to the GUI display.

text is the string that is inserted into the editbox

Top X, Top Y, Bottom X and Bottom Y define a box in which the editbox is drawn

id specifies a unique numerical reference for the editbox so events can be identified as coming from this object

border specifies whether the object has a border drawn around it. acceptable values are: -

IRR_GUI_NO_BORDER IRR_GUI_BORDER

password specifies whether the editbox is a password field that hides the text typed into it. acceptable values are: -

IRR_GUI_PASSWORD IRR_GUI_NOT_PASSWORD

Parent defines the parent object of this window. This can be ommited if the object has no parent.

Example

editboxObject = IrrAddEditBox("My String", 16,16,96,32, 120, IRR_GUI_BORDER, IRR_GUI_NOT_PASSWORD, windowObject)

IrrAddCheckBox

Syntax

IrrAddCheckBox (text as wstring, horizontal as integer, Top X as integer, Top Y as integer, Bottom X as integer, Bottom Y as integer, id as integer, checked as integer, parent as IRR_GUI_OBJECT) as IRR_GUI_OBJECT

Description Add a checkbox object to the GUI display.

text is the string that is used to label the checkbox

Top X, Top Y, Bottom X and Bottom Y define a box in which the checkbox is drawn

id specifies a unique numerical reference for the checkbox so events can be identified as coming from this object

checked specifies whether the object starts in the checked state. acceptable values are: -

IRR_OFF IRR_ON

Parent defines the parent object of this window. This can be ommited if the object has no parent.

Example

checkboxObject = IrrAddCheckBox("My Checkbox", 16,16,96,32, 120, IRR_OFF, windowObject)

IrrCheckCheckBox

Syntax

IrrCheckCheckBox (checkbox as IRR_GUI_OBJECT, checked as integer)

Description

Set the checked state of a checkbox.

checkbox defines the checkbox GUI object to check or uncheck.

checked specifies whether the object starts in the checked state. acceptable values are: -

IRR_OFF IRR_ON

Example

IrrCheckCheckBox(checkboxObject, IRR_ON)

IrrAddImage

Syntax

IrrAddImage (texture as IRR_TEXTURE,horizontal as integer, X as integer, Y as integer, useAlpha as integer, id as integer, parent as IRR_GUI_OBJECT) as IRR_GUI_OBJECT

Description Add an image object to the GUI display.

texture is a loaded texture object that is to be displayed

X, Y define a position at which the image is drawn

useAlpha specifies whether the alpha channel of the texture is to be used. acceptable values are: -

IRR_IGNORE_ALPHA IRR_USE_ALPHA

id specifies a unique numerical reference for the image so events can be identified as coming from this object

Parent defines the parent object of this window. This can be ommited if the object has no parent.

Example

imageObject = IrrAddImage(texture, 16,16, IRR_IGNORE_ALPHA, 120, windowObject)

IrrAddFileOpen

Syntax

IrrAddFileOpen (title as wstring, id as integer, checked as integer, modal as integer, parent as IRR_GUI_OBJECT) as IRR_GUI_OBJECT

Description

Open a modal file open dialog so that a file can be selected.

title is the string that is displayed in the titlebar of the file selector window.

id specifies a unique numerical reference for the button so events can be identified as coming from this object

Modal determines if the window locks out the rest of the interface until it is closed: -

IRR_GUI_MODAL IRR_GUI_NOT_MODAL

Parent defines the parent object of this window. This can be ommited if the window has no parent.

Example

```
fileOpenObject = IrrAddFileOpen( "Select a bitmap", 120,
IRR_GUI_MODAL, windowObject )
```

IrrGetLastSelectedFile

Syntax

IrrGetLastSelectedFile (fileopenobject as IRR_GUI_OBJECT, checked as integer)

Description Get the last file name selected from a file selection dialog.

E

Example fileName = IrrGetLastSelectedFile()

Wrapper Structure Defintions

IRR_MOUSE_EVENT

action as uinteger

Action determines which mouse action took place and can be one of the following values: -

IRR_EMIE_LMOUSE_PRESSED_DOWN

IRR EMIE RMOUSE PRESSED DOWN

IRR_EMIE_MMOUSE_PRESSED_DOWN

IRR_EMIE_LMOUSE_LEFT_UP IRR EMIE RMOUSE LEFT UP

IRR_EMIE_MMOUSE_LEFT_UP

IRR_EMIE_MOUSE_MOVED

IRR_EMIE_MOUSE_WHEEL

delta as single This defines the amount of movement of the mouse wheel.

x as integer **y** as integer These define the screen co-ordinates at which the event took place.

IRR_KEY_EVENT

key as uinteger The scan code for the key

direction as uinteger Whether the key moved up or down, this can be either of: -IRR_KEY_UP IRR_KEY_DOWN

flags as uinteger Bits are set in this parameter to specify whether the shift or control key was

IRR_PARTICLE_EMITTER

min_box_x as single
min_box_y as single
min_box_z as single
max_box_x as single
max_box_y as single
max_box_z as single
These six parameters define a box in space inside which the position of a
particle is randomly created.
direction_x as single
direction_y as single
direction_z as single
These three parameters define a direction into which the particles will be
ejected as the animation plays

min_paritIcles_per_second as uinteger **max_paritIcles_per_second** as uinteger A range defining the minimum and maximum number of particles that will be created each second.

min_start_color_red as integer min_start_color_green as integer min_start_color_blue as integer max_start_color_red as integer max_start_color_green as integer max_start_color_blue as integer Although particles can be textured by texturing the particle system node, these can be used to apply a range that tints the color of the particles.

min_lifetime as uinteger **max_lifetime** as uinteger How long the partilce will live, long lifespans can create very large numbers of particles

min_start_sizeX as single

min_start_sizeY as single

max_start_sizeX as single

max_start_sizeY as single

The minimum and maximum start sizes for the particles.

max_angle_degrees as integer

The maximum number of degrees that the ejected particles will deviate from the defined direction

IRR_VERT

A vertex is a point is space that also defines a number of properties that can be applied to the corner of a triangle.

x as single **y** as single **z** as single The 3D position of the vertex

normal_x as single
normal_y as single
normal_z as single
The normal direction of the vertex

vcolor as uinteger The 32bit ARGB color of the vertex

texture_x as single
texture_y as single
The 2 dimensional co-ordinate of the vertex when it is mapped to an applied
texture (0 to 1)

IRR_VECTOR

x as single **y** as single **z** as single A point that can be use for co-ordinates, directions or speeds.

61	

t-----

Wrapper Library for Irrlicht

Release Notes



Known Issues

- Currently the material type IRR_EMT_TRANSPARENT_ADD_ALPHA_CHAN: when applied to grass will only work when using the material type IRR_EMT_FOUR_DETAIL_MAP on the terrain.
- Newton Physics only uses convex hulls for collisions at the moment, this needs to be optimised so that simple boxes, spheres and cylinders can also be used to improve the performance.
- The surface of Tiled Terrains are now inverted in the X axis, this has been inherited from the Irrlicht Terrain node it has been derived from. For compatibility this will now be the normal orientation for a Terrain Tile.
- Wrappers built with g++ 4.4.0 are exhibiting some mathematical errors that can be seen in example 91. The supplied binary is built with g++ 3.4.5 and MSVC 6 and does not exhibit these errors.
- The supplied Linux binaries distributed in a seperate package are built on Fedora 11 with g++ 4.4.0 using the mesa libraries. They have only been tested on Fedora 11. Any project using this wrapper should rebuild their own binaries.
- Certain cards have been shown to exhibit mipmap

texture corruption when used with textures that have dimentions that are not a power of two. It is unclear whether this is a problem with the Wrapper, Irrlicht or the Graphics Drivers. It is therefore recommended that power of two textures are always used (e.g: 128x128, 512x256, etc ...)

v0.7.7

Added a new LOD manager that changes the meshes used by child mesh objects at particular distances and can fade the node in and out.

Added a new Billboard Group node that is an optimised collection of billboard meshes that can be used for custom particle effects.

Added a new Fade animator that fades, scales and eventually deletes the node from the scene.

Added a new Electrical Bolt effect node that simulates the effect of an electrical discharge. Scene Node created by **Sudi** with extensions from **Trivtn**

Added a new Beam effect node that simulates a laser or tracer beam type effect. Scene Node created by **Gaz Davidson (Bitplane)**

Added IrrAddBoltSceneNode to create an electrical bolt effect.

Added IrrSetBoltProperties to set the properties of an electrical bolt effect

Added IrrAddBeamSceneNode to create a beam effect.

Added IrrSetBeamSize to set the size of a beam effect.

Added IrrSetBeamPosition to set the start and end position of a beam effect.

Added IrrAddFadeAnimator attach a fade, scale and delete animator to a node.

Added IrrAddLODManager to create an LOD management system.

Added IrrAddLODMesh to add a level of detail to a LOD management system

Added IrrSetLODMaterialMap to allow the assignment of materials that are used as substitues for a nodes normal material to face them out over time

Added IrrSetNodeMesh to set the mesh of a node

Added IrrSetBillBoardColor to set the colour of a standard billboard

Added IrrSetBillBoardSize to set the size of a standard

billboard

Added IrrAddBillBoardGroupToScene to add a billboard group management system

Added IrrAddBillBoardToGroup to add a billboard to a billboard group

Added IrrAddBillBoardByAxisToGroup to add a billboard fixed to an axis to a billboard group

Added IrrRemoveBillBoardFromGroup to remove a billboard from a billboard group

Added IrrBillBoardGroupShadows to shade billboards in a billboard group with lighting from a particular direction

Added IrrGetBillBoardGroupCount to retrieve the count of the number of billboards in a billboard group

Added IrrBillBoardForceUpdate to force a billboard group to update their orientations to the camera.

Added IrrStartAdvanced to provide an advanced start up function that allows for the use of anti-aliasing and high precision floating point calculations. Contributed by Agamemnus. Added IrrBeginSceneAdvanced to provide an advance begin scene function providing control over whether the back buffer and buffer are erased. Contributed by Agamemnus.

Added IrrSetRenderTarget to provide control for switching the output of the drawing operations perminantly to a texture. Contributed by Agamemnus.

Added IrrGet2DPositionFromScreenCoordinates to get a coordinate on a 2D plane from a ray cast through a camera. Contributed by Agamemnus.

Added IrrSetNodeColorByVertex to change the way vertex color effects all materials in a node. Contributed by Agamemnus.

Added IrrSetNodeAmbientColor to set the ambient color of all materials associated with a node. Contributed by Agamemnus.

Added IrrSetNodeDiffuseColor to set the diffuse color of all materials associated with a node. Contributed by Agamemnus.

Added IrrSetNodeSpecularColor to set the specuilar color

of all materials associated with a node. Contributed by Agamemnus.

Added IrrSetNodeEmissiveColor to set the emissive color of all materials associated with a node. Contributed by Agamemnus.

v0.7.6

Added IrrGetScreenSize to get the size of the screen. Contributed by Agamemnus.

Added IrrGetAbsoluteMousePosition to get the absolute position of the mouse. Contributed by Agamemnus.

Added IrrGetNodeScale to determine the scaling applied to a node.

Added IrrGetScreenSize to get the size of the screen.

Added IrrIsFullscreen to check if the application is fullscreen mode.

Added IrrIsWindowActive to check if the window is active.

Added IrrIsWindowFocused to check if the window has focus.

Added IrrIsWindowMinimized to check if the window is minimised.

Added IrrMaximizeWindow to maximise the window.

Added IrrMinimizeWindow to minimise the window.

Added IrrRestoreWindow to restore the size of the window.

Added IrrResizableWindow to make a window resizable by the user.

Changes to irrlicht_transforms.bi by Agamemnus to include the following things: -

- Define multikeyold and getmouseold before undefining multikey and getmouse. Now people can still use the "old" multikey functions if they close the Irrlicht window later on in their program.
- Added "continue do" to all mouse select case cases.
- Added a clip test.

Reworked IrrlichtNewton.bi to provide a simple interface and basic example function on Newton collision. The origonal include file is supplied as "IrrlichtNewtonGamePhysics.bi"

Added IrrlichtODE.bi to provide almost identical physics support as is supplied for IrrlichtNewton.bi only for ODE physics. This is all based on the ODE include files contributed to FreeBASIC by D.J.Peters.

Added the Batching Meshes module from **Gaz Davidson** (**Bitplane**) if your scene contains hundreds of static scenenodes this technique can greatly increase the framerate by batching them all together in one set of geometry. You can realise as high as a **40 times** speed increase!

Added IrrCreateBatchingMesh to create an empty batching mesh that is a collection of other meshes and used to reduce node counts in a scene thereby increasing framerate.

Added IrrAddToBatchingMesh to add a standard static mesh to a batch of meshes.

Added IrrFinalizeBatchingMesh to complete the batching mesh once all meshes have been added and recover a standard mesh object.

Added IrrSetMeshMaterialTexture to set the texture image associated with a mesh.

Added IrrGetMeshVertexMemory to get the memory for the array of vertex data structures in the Irrlicht Engine.

Added the "**XEffects - Reloaded**" system produced by **BlindSide** from the irrlicht community, this contains several shaders that can have a massive visual impact on the realism of your scene. The Dynamic Shadow casting lights probably have the biggest impact. of all of the shaders introducing quality realistic lighting into the scene.

Added IrrXEffectsStart to start the XEffects GLSL and HLSL shader support system.

Added IrrXEffectsEnableDepthPass to enable a depth rendering pass in XEffects used with the SSAO and water shaders.

Added IrrXEffectsAddPostProcessingFromFile to add a GLSL or HLSL shader to the engine from a file.

Added IrrXEffectsSetPostProcessingUserTexture to set the texture associated with certain XEffects.

Added IrrXEffectsAddShadowToNode to add a dynamic shadow to a node.

Added IrrXEffectsRemoveShadowFromNode to remove a dynamic shadow from a node.

Added

IrrXEffectsExcludeNodeFromLightingCalculations to exclude a node from lighting calculations in the XEffects rendering passes.

Added IrrXEffectsAddNodeToDepthPass to add a node to XEffects depth pass rendering.

Added IrrXEffectsSetAmbientColor to set the ambient level of lighting the the XEffects shaders.

Added IrrXEffectsSetClearColor to set the background color when using XEffect post processing shaders.

Added IrrXEffectsAddShadowLight to add a shadow casting XEffects light.

Added IrrXEffectsSetShadowLightPosition to set the position of an XEffects shadow casting light.

Added IrrXEffectsGetShadowLightPosition to get the position of an XEffects shadow casting light.

Added IrrXEffectsSetShadowLightTarget to set the target of an XEffects shadow casting light.

Added IrrXEffectsGetShadowLightTarget to get the

target of an XEffects shadow casting light.

Added IrrXEffectsSetShadowLightColor to set the color of an XEffects shadow casting light.

Added IrrXEffectsGetShadowLightColor to set the color of an XEffects shadow casting light.

Tidyied up styling of HTML documentation

Moved Lighting discussion to the IrrlichtWrapper Website.

v0.7.5

Remove the unsupported IrrLockOpenGLTexture from the reference and corrected heading for IrrUnlockImage.

Added command IrrGetNodeMesh to get the mesh associated with a scene node contributed by agamemnus

Modified IrrStop to close the device, this now means that resources are tidied up correctly and a new device can even be opened to change resolution.

Modified the handling of Keyboard, Mouse and GUI events so that all events are recieved and the GUI does not block the other two.

Removed IrrGUIEvents as it is no longer needed.

Added IrrRenameMesh to allow a loaded mesh to be renamed

Added IrrScaleMesh to allow for the uniform scaling of a mesh without effecting its vertex normals.

Added IrrDraw2DImageElementStretch to copy an image to the screen while scaling it too. Contributed by Eric

(The Car)

Added IrrSetNodeTriangleSelector to attach a triangle selector to a specific node for multiple nodes. Recommended by Agamemnus.

Changed IrrSetMeshVertexCoords, IrrSetMeshVertexCoords and IrrSetMeshVertexSingleColor to include an extra index so for example a group [0] to [0] would include one vertex.

Added IrrGetTextureInformation to get dimentions, pitch and color format of a texture.

Added IrrGetImageInformation to get dimentions, pitch and color format of an image.

Added IrrSetNodeRotationPositionChange to allow for the incremental rotation and direction dependant incremental position change of a node while returning information that can be used to attach a camera and locate other objects and effects. This code is based on examples from Chev and also in large part the cockpit example functions by Arras.

Added IrrGetMeshBoundingBox to get the bounding box of a mesh. Contributed by Agamemnus.

Added IrrGetNodeTransformedBoundingBox to get the tranformed absoloute position of a nodes bounding box. Contributed by Agamemnus.

Changed IrrAddNodeShadow to add an optional parameter allowing a low resoloution mesh to be used in shadow calculation.

v0.7.4

Changed IrrMaterialVertexColourAffects to IrrMaterialVertexColorAffects

Changed IrrSetNodeAnimationSpeed to take a speed parameter of type floating point to conform with Irrlicht signiture changes.

Changed IrrSetNodeAnimationFrame to take a frame parameter of type floating point to conform with Irrlicht signiture changes.

Added function IrrGetDistanceBetweenNodes to get the distance between two nodes efficiently.

Added function IrrAreNodesIntersecting to determine if the bounding boxes of two nodes are intersecting.

Added function IrrIsPointInsideNode to determine if a point is inside the bounding box of a node.

Added function IrrGetCollisionResultPosition to add eplisoid collision management supplied by 'The Car'.

Corrected IrrGetCollisionGroupFromMesh to return a

triangle selection.highlighted by super_castle

Updated Newton functions provided by SiskinEDGE to support mesh buffer.

Corrected missing parameters in IrrSetMeshVertexSingleColor highlighted by agamemnus

Updated IrrRemoveCollisionGroup to allow it to take a scene node parameter that lets the selector be removed from an associated node.

Updated IrrGetCollisionGroupFromComplexMesh and IrrGetCollisionGroupFromBox so the created selector is attached to the scene node they are created from.

Added function IrrGet3DPositionFromScreenCoordinates to get a 3D position on a plane from a set of screen coordinates, contributed by agamemnus

Included correction to CParticleEmissionEffector constructor used in IrrAddStopParticleAffector to allow the stop time to be set correctly. Contributed by The Car

Added example 91_Example_DistanceAndCollision.bas to demonstrate new distance and collision functions.

Added example

92_Example_MovingEntitiesByCollision.bas to demonstrate moving entities around a scene through collision.

Added example 93_Example_3DPositionFromScreen.bas to demonstrate getting a point on a plane from screen co-ordinates.

Corrected example 50_Example_Fading_Nodes_Out.bas to correctly demonstrate fading nodes.

Resolved all compilation warnings at warning level 3 when built under msvc6 and g++3.4.5

v0.7.3

Added in new video feature queries supported by Irrlicht for Vertex Buffer Object, Alpha to Coverage, Color Masks, Multiple Render Targets, Blend settings for Multiple Render Targets and Geometry Shaders.

Added IrrSetMaterialBlend for setting the Blend settings of materials associated with the ONETEXTURE_BLEND material.

Added IrrAddBillboardTextSceneNode for adding a billboard with text on it that can be used as a scaling label in a 3D environment.

Added IrrSetTime for setting the irrlicht animation time in the system.

Updated example 23_Example_Video_Features.bas to display new feature capabilities.

Updated example 81_Example_GLSL_Shader_Materials.bas to display a range of GLSL materials.

Added example 86_Example_Screenshot_to_texture.bas

to demonstrate capturing screenshots to textures.

Added example 87_Example_MultiPass_Rendering.bas to demonstrate blending several rendering passes to the display.

Added example 88_Example_Billboard_Text.bas to demonstrate adding billboard text and attaching it to a node.

Added example 89_Example_Orthagonal_Camera.bas to demonstrate the use of orthagonal cameras

Added example 90_Example_Collision_Point.bas to demonstrate a new ray collision test feature.

Added IrrDisableFeature for disabling video driver features.

Added IrrGetNodeBoundingBox to get the bounding box of a node. supplied by Agamemnus

Added IrrSetMaterialLineThickness to set the material line thickness of vertex drawing operation. supplied by Agamemnus

Added IrrSetMeshVertexSingleColor to set groups of

verticies in a mesh to a uniform color. supplied by Agamemnus

Added IrrGetCollisionGroupFromMesh to creates a collision group from the triangles in a mesh.

Added IrrRemoveCollisionGroup to remove a collision group freeing the memory it uses.

Added IrrGetNodeAndCollisionPointFromRay to detect a collision between a ray and scene node objects that have had collision groups created for them.

Added IrrDisableFeature to disable particular video features.

Added IrrGetMeshFrameCount to get the number of frames in an animated mesh

Added IrrGetMeshBufferCount to get the number of mesh buffers in a mesh.

Added IrrSetCameraOrthagonal to set the camera to display an Orthagonal view instead of the default persepective view.

Updated IrrGetMeshIndexCount, IrrGetMeshIndices,

IrrSetMeshIndices, IrrGetMeshVertexCount, IrrGetMeshVertices, IrrSetMeshVertices, IrrSetMeshVertexColors, IrrSetMeshVertexCoords and IrrSetMeshVertexSingleColor to work from individual mesh buffers instead of the entire mesh as a single object.

Corrected IrrSetMeshVertices and IrrSetMeshVertexCoords to correctly set the bounding box of the mesh they modify.

v0.7.2

Changed the IrrGetScreenShot command to capture a portion of the screen to texture rather than the whole screen to an image.

Corrected a bug where mouse events were not being passed to the active camera when keyboard and mouse is being captured.

Corrections for bugs in passing variables to GLSL materials.

v0.7.1

Updated documentation with new commands

Updated IrrlichtWrapper to compile with Irrlicht version 1.7.1

Updated IrrlichtWrapper to compile with g++ upto version 4.4.1

Added IrrSetMeshVertexColors for setting the color of verticies supplied by Agamemnus

Added IrrSetMeshVertexCoords for setting the coordinates of verticies supplied by Agamemnus

Added IrrAddEmptySceneNode to add an empty scene node to the scene supplied by Agamemnus

Added IrrGetCameraUpDirection to get the up direction of the active camera supplied by Agamemnus

Added IrrGetScreenShot to grab a screenshot into an image

Added IrrRemoveImage to remove an image from

memory

Added IrrMaterialVertexColourAffects to set the material property that vertex colour effects

Added IrrGUIClear to clear all GUI objects from the screen

Added IrrGUIRemove to remove a specific GUI object

Added IrrGUIGetText to get the text associated with a GUI object

Added IrrGUISetText to set the text associated with a GUI object

Added IrrAddWindow to add a blank window GUI object to the display

Added IrrAddButton to add a clickable button GUI object to the display

Added IrrAddScrollBar to add a scrollbar GUI object to the display

Added IrrAddListBox to add a listbox GUI object to the display

Added IrrAddListBoxItem to add items to a listbox GUI object

Added IrrInsertListBoxItem to insert items into specific positions in a listbox GUI object

Added IrrRemoveListBoxItem to remove a specific item from a listbox GUI object

Added IrrSelectListBoxItem to select a specific item in a listbox GUI object

Added IrrAddEditBox to add an editable editbox GUI item

Added IrrAddImage to add a 2D image GUI item

Added IrrAddCheckBox to add a clickable checkbox GUI item

Added IrrCheckCheckBox to set the check state of a checkbox GUI item

Added IrrAddFileOpen to add a file open GUI item

Added IrrGUISetFont to set the font associated with the

GUI

Added IrrGUISetColor to set the colour of the GUI

Added IrrGUIEvents to start or end capturing GUI events

Added IrrGUIEventAvailable to check if there is a GUI event available

Added IrrReadGUIEvent to read a GUI event

Added IrrGetLastSelectedFile to get the last select file from a GUI file selection object

Added IrrSetViewPort to allow rendering to be confined to an area of the screen for split screen effects

Added IrrSetCameraAspectRatio to allow the aspect ratio of a camera to be changed primarily for split screen effects

Updated example 36 materials to work with Irrlicht 1.7.1

Added examples to demonstrate a GLSL material, GUI, Split Screen, Freetype fonts and Embedded OpenGL commands. Added example 85_Example_Split_Screen.bas to demonstrate splitscreen rendering.

v0.7

Newton Physics Engine support for convex hulls by SiskinEDGE

Modified to work with Irrlicht v1.6

Support for hardware accelerated meshes allowing static models to .

Support for Crossed grass, correction to reduce grass 'popup'

Particle systems now have minimum and maximum particle size properties in their configuration structures

v0.6.2

Corrections to allow compilation under linux without modification

Captured keyboard commands are now also sent to the active camera.

Correction to example 44 to prevent a crash under build for Direct3D

Corrections to build for DirectX to support all 236 commands in the library

v0.6.1

Corrected examples to not use litteral paths thanks AlexZ

Added column support to push particle affector

Added IrrSetColumnDistanceOfEffect to allow the push affector columnn effect to be dynamically changed

Added IrrCreateImage to create blank images

Added IrrLockImage to get the pixels for an image

Added IrrUnlockImage to release the pixels for an image

Added IrrGetSphericalTerrainSurfacePosition for getting coordinates from a sphereical terrains face number and a logical X,Z coordinate

Added IrrGetSphericalTerrainSurfacePositionAndAngle for getting coordinates from a sphereical terrains face number and a logical X,Z coordinate, this also returns the angle of the point at the surface, useful for placing objects on the spheres surface.

Added IrrGetSphericalTerrainLogicalSurfacePosition for

getting the logical face number and X,Z coordinates from normal spacial coordinates. (This is not 100% accurate yet) (not supported in DirectX in this release)

Added IrrAddMayaCamera for adding a maya style camera to the view you can click and drag to move, revolve and zoom this camera. Suggested by Alvaro

Updated documentation with new commands

Added examples 70 to 76 to demonstrate features

Minor correction to example 12 thanks Alvaro

Significant improvement to Example 14 to show proper manual camera control

Corrected IrrSetCameraUpDirection in the manual thanks AlexZ

v0.6

Included a simple hacha.jpg file to correct examples 21, 41 and 43

Added IrrAddClouds to create a clouds node by Gaz Davidson and Nikolaus Gebhardt

Added IrrAddGrass to create a patch of grass across a terrain by Gaz Davidson

Added IrrSetSkyDomeColor for tinting the texture on a skydome

Added IrrAddStopParticleAffector to stop particle flows

Added IrrSetSkyDomeColor for setting the color of verticies in a skydome

Added IrrAddZoneManager for adding automatic node management

Added IrrGetImage for loading image files that do not use video memory

Added IrrAddTerrainTile for adding terrain objects that

can be tiled!

Added IrrScaleTileTexture for scaling tiled terrain objects

Added IrrAttachTile for attaching tilable terrain objects together

Added IrrSetTileColor for loading the vertex color of terrain tiles

Added IrrSetTileStructure for loading the height and vertex color of terrain tiles

Added IrrSetGrassDensity for changing grass density

Added IrrSetGrassWind for changing the strength of wind affecting the grass

Added IrrSetZoneManagerAttachTerrain for attaching a terrain to a zone

Added IrrRevolveCamera for 6DOF (six degrees of freedom) camera motion for spacecraft and aircraft

Added IrrSetMousePosition for moving the mouse on the screen

Added IrrTransparentZWrite for ordering transparent objects. This causes a performance hit but can correct situations where transparent objects appear in the wrong order on the screen

Added IrrGetTerrainTileHeight to get the height of a point on a tile terrain

Added IrrGetTerrainHeight to get the height of a point on a normal terrain

Added IrrAddLensFlare to create a camera lens flare effect

Added IrrSetFlareScale to set the scale of the lens flare effect

Added IrrSetSkyDomeColorBand to add a band of color to a skydome

Added IrrSetSkyDomeColorPoint to add a circle of color radiating out from a point to the vertices of a skydome

Added IrrSetZoneManagerProperties to set the properties of a zone management object

Added IrrSetNodeName to set the name of a node

Added IrrGetNodeName to get the name of a node

Added IrrGetRootSceneNode to get the root object within which all other nodes are contained

Added IrrGetChildCollisionNodeFromRay to get a child object hit by a ray

Added IrrSetZoneManagerBoundingBox to get the bounding box of a zone managment object

Added IrrGetChildCollisionNodeFromPoint to get a child object contained in a specific location

Added IrrGetNodeFirstChild to get the first child object of a specific object

Added IrrIsNodeLastChild to check if an object is the last child object of a specific object

Added IrrGetNodeNextChild to get the next child object of a specific object

Added IrrGetNodeAbsolutePosition to get the absolout position of a node taking into account the positions of all of its parents Added IrrSetNodeParent to set the parent node of a specific node

Added IrrSetCameraUpAtRightAngle to adust the camera up angle so that it is at right angles to the camera vector

Added IrrAddSphericalTerrain to create a spherical terrain object for planets

Added IrrScaleSphericalTexture to scale the textures of a spherical terrain object

Added IrrAddStopParticleAffector to create a particle affector that stops the flow of particles from a specific emitter

Added IrrAddParticlePushAffector to create a particle affector that pushes particles in particular directions

Added IrrAddColorMorphAffector to create a particle color morph affector that changes the color of a particle over its lifetime

Added IrrSetFurthestDistanceOfEffect to set the furthest distance that the push affector has any influence

Added IrrSetNearestDistanceOfEffect to set the nearest distance that the push affector has any influence

Added IrrSetCenterOfEffect to set the center of the effect of a push affector

Added IrrSetStrengthOfEffect to set the strength of a push affector

Added IrrAddSplineAffector to create a spline path affector for particles

v0.5.1

Included a new parameter into IrrStart that allows the wrapper to enable Irrlichts vertical syncronisation to prevent 'tearing' of the picture by syncronising the refresh of the display. (suggested by Daiwa)

V0.5

Eponasoft contributed support for skydomes with IrrAddSkyDomeToScene

Eponasoft contributed support for water nodes with IrrAddWaterSurfaceSceneNode

Eponasoft contributed support for hill planes with IrrAddHillPlaneMesh

Eponasoft contributed support for loading IrrEdit scenes with IrrLoadScene

Added support for saving the current scene to a file that can be loaded into irrEdit with IrrSaveScene.

Added support for finding a node in the scene by its ID with IrrGetSceneNodeFromId

Added support for finding a node in the scene by its name with IrrGetSceneNodeFromName

Support for adding a simple cube scene node

Support for adding a simple Sphere scene node

Added support for the Irrlicht call to set texture creation flags with IrrSetTextureCreationFlag

Added support for the Irrlicht call to draw a 3D line into the display with IrrDraw3DLine

Added support for the Irrlicht call to create a texture that is suitable for the scene manager to use as a surface to which it can render its 3d scene with IrrCreateRenderTargetTexture. Each the dimentions must be a power of two for example 128x128 or 256x256

Added support for the Irrlicht call to draw scene manager objects to a texture surface with IrrDrawSceneToTexture. The texture must have been created with a call to IrrCreateRenderTargetTexture

Added support for the Irrlicht call to create a blank texture with IrrCreateTexture

Added support for the Irrlicht call to lock the texture and returns a pointer to the pixels with IrrLockTexture

Added support for the Irrlicht call to unlock the texture with IrrUnlockTexture, presumably after it has been modified and recreate the mipmap levels Added support for the Irrlicht call to remove a texture from memory with IrrRemoveTexture freeing the resources

Added support for the Irrlicht query feature command. The video card can now be queried to see which video card features it supports.

Added support for the Irrlicht call to add a mesh to the scene as a static object with IrrAddStaticMeshForNormalMappingToScene, the mesh is altered so that it is suitable for the application of a Normal or Parallax mapping material, at this time any animation information is lost.

Added support for Irrlicht call to make a normal map from a greyscale bump map texture with IrrMakeNormalMapTexture

Added support for Irrlicht call to Get the number of materials associated with a node with IrrGetMaterialCount

Added support for Irrlicht call to Get the material associated with the node at the particular index with IrrGetMaterial Added support for Irrlicht call to Set how shiny the material is with IrrMaterialSetShininess the higher the value the more defined the highlights

Added support for Irrlicht call to set the color of specular highlights on the object with IrrMaterialSetSpecularColor

Added support for Irrlicht call to set the color of diffuse lighting on the object with IrrMaterialSetDiffuseColor

Added support for Irrlicht call to set the color of ambient light reflected by the object with IrrMaterialSetAmbientColor

Added support for Irrlicht call to set the color of light emitted by the object with IrrMaterialSetEmissiveColor

Added support for Irrlicht call to set material specific parameter with IrrMaterialSetMaterialTypeParam

Added a materials interface with a series of functions for adding GPU Shader programs for modern graphics cards. This interface is in an early phase of development and requires more work.

Added support for Irrlicht calls for creating Materials

based upon GPU programs for cards that support the feature : IrrAddHighLevelShaderMaterial, IrrAddHighLevelShaderMaterialFromFiles, IrrAddShaderMaterial and IrrAddShaderMaterialFromFiles

Added support for Irrlicht calls for creating constants to materials with: IrrCreateNamedVertexShaderConstant, IrrCreateNamedPixelShaderConstant, IrrCreateAddressedVertexShaderConstant and IrrCreateAddressedPixelShaderConstant

IrrGetMS3DJointNode and IrrGetDirectXJointNode are now replaced with a universal IrrGetJointNode call.

Added support for Irrlicht calls for setting the current frame number being played in the animation with IrrSetNodeAnimationFrame

Added support for Irrlicht calls for setting the time in seconds across which two animation frames are blended with IrrSetTransitionTime. This can be used to smothly animate a model between two poses where no animation data exists.

Added support for Irrlicht calls for animating the mesh based on the position of the joints, this should be used at the end of any manual joint operations including blending and joints animated using IRR_JOINT_MODE_CONTROL and IrrSetNodeRotation on a bone node with IrrAnimateJoints

Added support for Irrlicht calls for setting the animation mode of joints in a node with IrrSetJointMode allowing them to be blended or controlled programatically.

Added support for Irrlicht call to write the first frame of the supplied animated mesh out to a file using the specified file format with IrrWriteMesh

Added support for Irrlicht calls for creating a particle emitter for an animated mesh scene node with IrrCreateAnimatedMeshSceneNodeEmitter

Added support for Irrlicht calls for creating a point attraction affector with IrrAddParticleAttractionAffector. This affector modifies the positions of the particles and attracts them to a specified point at a specified speed.

Added support for Irrlicht calls for creating a rotation affector with IrrCreateRotationAffector. This affector modifies the positions of the particles and attracts them to a specified point at a specified speed per second. Added support for Irrlicht calls for setting direction the emitter emits particles with IrrSetParticleEmitterDirection.

Added support for Irrlicht calls for setting minimum number of particles the emitter emits per second with IrrSetParticleEmitterMinParticlesPerSecond.

Added support for Irrlicht calls for setting maximum number of particles the emitter emits per second IrrSetParticleEmitterMaxParticlesPerSecond.

Added support for Irrlicht calls for setting minimum starting color for particles with IrrSetParticleEmitterMinStartColor.

Added support for Irrlicht calls for setting maximum starting color for particles IrrSetParticleEmitterMaxStartColor.

Added support for Irrlicht calls for enabling or disabling an affector with IrrSetParticleAffectorEnable.

Added support for Irrlicht calls for altering the fadeout affector changing the fade out time with IrrSetFadeOutParticleAffectorTime. Added support for Irrlicht calls for altering the fadeout affector changing the target color IrrSetFadeOutParticleAffectorTargetColor.

Added support for Irrlicht calls for altering the direction and force of gravity with IrrSetGravityParticleAffectorDirection.

Added support for Irrlicht calls for setting the time in milliseconds when the gravity force is totally lost and the particle does not move any more with IrrSetGravityParticleAffectorTimeForceLost.

Added support for Irrlicht calls for setting whether or not this will affect particles in the X direction with IrrSetParticleAttractionAffectorAffectX.

Added support for Irrlicht calls for setting whether or not this will affect particles in the Y direction with IrrSetParticleAttractionAffectorAffectY.

Added support for Irrlicht calls for setting whether or not this will affect particles in the Z direction with IrrSetParticleAttractionAffectorAffectZ.

Added support for Irrlicht calls for setting whether or not

the particles are attracting or detracting with IrrSetParticleAttractionAffectorAttract.

Added support for Irrlicht calls for setting the point that particles will attract to with IrrSetParticleAttractionAffectorSetPoint.

Added support for Irrlicht calls for setting the point that particles will rotate about with IrrSetRotationAffectorSetPivotPoint.

Added support for Irrlicht calls for clearing all meshes that are held in the mesh cache but not used anywhere else with IrrClearUnusedMeshes

Added support for Irrlicht calls for removing all nodes in a scene with IrrRemoveAllNodes. This is useful for example for clearing the scene between levels.

Added support for Irrlicht calls for saving a screenshot out to a file with IrrSaveScreenShot

Added support for Irrlicht calls for setting the Ambient color emitted by the light with IrrSetLightAmbientColor

Added support for Irrlicht calls for setting the the light strength fading over distance with IrrSetLightAttenuation

Added support for Irrlicht calls for setting whether the light casts shadows with IrrSetLightCastShadows

Added support for Irrlicht calls for setting the Diffuse color emitted by the light with IrrSetLightDiffuseColor

Added support for Irrlicht calls for setting the lights strength's decrease between Outer and Inner cone with IrrSetLightFalloff

Added support for Irrlicht calls for setting the angle of the spot's inner cone with IrrSetLightInnerCone

Added support for Irrlicht calls for setting the angle of the spot's outer cone IrrSetLightOuterCone

Added support for Irrlicht calls for setting the Radius of light with IrrSetLightRadius. Everything within this radius be be lighted.

Added support for Irrlicht calls for setting the Specular color emitted by the light with IrrSetLightSpecularColor

Added support for Irrlicht calls for setting the type of the light with IrrSetLightType

Adding lighting attributes has provided support to defining spot lights and distant light, the position and direction of these directional light sources can be changed by moving and rotation the light node.

61	

t-----

Wrapper Library for Irrlicht

Platform and Compiler Support



Overview

IrrlichtWrapper is supplied with three sets of files. MSVC versions, GCC versions and Linux Versions. The default version of the files are those compiled with MSVC, these provide support for OpenGL, Direct X 8, Direct X 9 and software rendering. The GCC and Linux versions support both OpenGL and Software rendering.

GCC Compiler

To use the GCC DLLs you need to rename the supplied files: -

- 1. Rename Irrlicht.dll to Irrlicht_ms.dll
- 2. Rename IrrlichtWrapper.dll to IrrlichtWrapper_ms.dll
- 3. Rename Irrlicht_gcc.dll to Irrlicht.dll
- 4. Rename IrrlichtWrapper_gcc.dll to IrrlichtWrapper.dll

You will now be able to build the IrrlichtWrapper with the GCC compiler and run the examples. You will not be able to use either Direct X rendering mode..

Linux Platform

The supplied linux library files libIrrlicht.so and libIrrlichtWrapper.so can be installed to your, /usr/lib directory, you will then be able to use the freebasic compiler to build and run the example files.

Linux support is limited at the moment the release is tested under Software Rendering on Fedora 11. The supplied libraries may provide reduced performanceand it is recommended that you rebuild both the Irrlicht Library and the IrrlichtWrapper library on your own system to utilise the OpenGL support that may be available on your system.