

Indigo SDK Documentation

By Nicholas Chapman

March 2008

Indigo Renderer and Indigo Renderer SDK License

License for the Indigo Renderer software package and Indigo Renderer SDK ("Indigo")

Terms and Conditions for Use

You may use Indigo for uncommercial use.

You may use Indigo for commercial use, with the following restrictions:

You may not install or use Indigo on a render-farm, where the use of, or access to, the render farm (and/or software installed on the renderfarm computers, and/or other services)

is sold to any third party, without an explicit written agreement for such usage with Nicholas Chapman.

'Use of Indigo on a render-farm' is defined as the use of the network rendering capability of Indigo on two or more computers networked together.

Terms and Conditions for distribution

You may not distribute or redistribute any or all parts of Indigo, or Indigo as a whole, without an explicit written agreement for such distribution with Nicholas Chapman.

(Note that this also applies to the Indigo dynamic link library available in the Indigo SDK)

(Indigo should only be available as a download from <http://www.indigorenderer.com>)

Nicholas Chapman

nick@indigorenderer.com

SDK Overview

The API documentation documents the interface exposed by the Indigo dynamic link library.

You can also refer to the example program included in the SDK, and also to the core Indigo documentation, which defines the scene definition XML in detail.

API Reference

indigoCreateInstance

This is the only procedure exported from the Indigo DLL. This procedure can be used to create an instance of a particular interface.

```
INDIGO_DLL_EXPORT indResult indigoCreateInstance(const indString interface_name,  
IndigoInterfaceBase** instance_out);
```

Parameters

interface_name

ASCII encoded, null-terminated string containing the name of the interface to be instantiated.

instance_out

Pointer to a pointer to an object derived from IndigoInterfaceBase. The referenced pointer is set to point to the newly created object.

Return Values

Returns one of the following error codes:

INDIGO_SUCCESS: Object of requested type was successfully created.

INDIGO_ERROR_UNKNOWN_INTERFACE: The requested interface is unknown and cannot be instantiated.

Extra information

Header file: IndigoDLL.h

IndigoInterface1

This is the interface for the main Indigo object. Using an instance of this object, the scene is defined, the rendering process is started, and the image is retrieved.

IndigoInterface1::initialise

Initialises the IndigoInterface1 object.

This method must be called before any other IndigoInterface1 methods.

```
indResult initialise(const indString base_indigo_directory_path)
```

Parameters

`base_indigo_directory_path`

Path to the directory that contains the Indigo DLL.

Should not contain a trailing slash.

Return Values

Returns one of the following error codes:

INDIGO_SUCCESS: Initialisation was successful.

INDIGO_ERROR_ALREADY_INITIALISED: `initialise()` has already been called.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::buildScene

Compiles the scene, building acceleration structures etc..

This method will block while scene compilation takes place.

```
indResult buildScene()
```

Parameters

Return Values

Returns one of the following error codes:

INDIGO_SUCCESS: Scene compilation was successful.

INDIGO_ERROR_CAMERA_NOT_SET: No camera was defined.

INDIGO_ERROR_TONEMAPPING_NOT_SET: No tonemapping was defined.

INDIGO_ERROR_NO_LIGHT_SOURCE_INSERTED: There are no light sources in the scene.

INDIGO_ERROR_NOT_INITIALISED: initialise() has not been called on the IndigoInterface1 object.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::startRendering

Starts the rendering process.

The rendering calculations themselves run in different threads.

```
void startRendering()
```

Parameters

Return Values

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::getBufferWidth

Returns the width of the image buffer, in pixels.

```
indUInt32 getBufferWidth()
```

Parameters

Return Values

Returns the image buffer width in pixels.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::getBufferHeight

Returns the height of the image buffer, in pixels.

```
indUInt32 getBufferHeight()
```

Parameters

Return Values

Returns the image buffer height in pixels.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::getBufferSize

Returns the size of the buffer, in Bytes, that needs to be passed to copyBuffer().

This size will be equal to width * height * 3

```
indUInt32 getBufferSize()
```

Parameters

Return Values

Returns the size of the buffer, in Bytes, that needs to be passed to copyBuffer().

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::copyBuffer

Tone-map the internal Indigo HDR buffer to a LDR, 8 bit per component sRGB format, and copy the LDR buffer to the buffer at the passed in address.

```
void copyBuffer(indUChar8* buffer)
```

Parameters

```
buffer
```

Must point to a buffer of size at least equal to the value returned by getBufferSize()

Return Values

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::startTerminateRender

Sets a flag that instructs the Indigo rendering threads to terminate as soon as possible.

Non-blocking.

```
void startTerminateRender()
```

Parameters

Return Values

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::isRenderTerminated

A query that returns if all the Indigo return threads have terminated.

Non-blocking.

```
indBool isRenderTerminated()
```

Parameters

Return Values

Returns a non-zero value if all render threads have terminated, zero otherwise.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::processXMLElement

Tells the Indigo DLL to process an XML element containing some part of the scene definition.

```
indResult processXMLElement(const indString work_path, const indString xml_element)
```

Parameters

```
work_path
```

Path relative to which all resource paths (texture paths, model paths, etc..) are taken.

Should not contain a trailing slash.

```
xml_element
```

Null terminated ASCII string containing exactly one XML element. (plus its children)

Return Values

INDIGO_SUCCESS: the element was parsed and processed successfully.

INDIGO_ERROR_OUT_OF_MEMORY: Ran out of memory.

INDIGO_ERROR_SYNTAX_ERROR: A syntax error was encountered while parsing the XML. Call `getLastErrorDescription()` for more information.

INDIGO_ERROR_GENERAL: Some other type of error occurred. Call `getLastErrorDescription()` for more information.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::startMeshDefinition

There are two ways to define triangle mesh geometry to be used in a scene – using embedded XML meshes and processXMLElement(), and using the lower level C++ interface.

The advantage of the lower level C++ interface is that it is much faster.

```
indResult startMeshDefinition(const indString mesh_name, indBool normal_smoothing)
```

Parameters

mesh_name

Null terminated ASCII string containing the name of the mesh. Must not have been used for a mesh already.

normal_smoothing

If true (non-zero), then the mesh will use interpolated smoothed normals. If false (zero), the mesh will use the geometric normals orthogonal to the triangle plane.

Return Values

INDIGO_SUCCESS: Call was successful.

INDIGO_ERROR_MESH_ALREADY_DEFINED: The mesh name has already been used.

INDIGO_ERROR_GENERAL: Some other type of error occurred.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::setMaxNumTexcoordSets

Meshes can use an arbitrary number of texture coordinate sets.

The number of texture coordinate sets is the number of texture coordinate 2-vectors per vertex.

This function must be called before vertices start being defined for this mesh.

It instructs the Indigo DLL how much space to reserve for vertex texture coordinates.

```
indResult setMaxNumTexcoordSets(indUInt32 max_num_texcoord_sets)
```

Parameters

max_num_texcoord_sets

Number of texture coordinate sets to use for the mesh.

Return Values

INDIGO_SUCCESS: Call was successful.

INDIGO_ERROR_GENERAL: A mesh is not currently being defined.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::addMaterialUsed

Adds a reference to a used material.

The index of the material for the mesh is the number of materials already added for this mesh. (e.g. The first material added has index 0 for this mesh)

```
indResult addMaterialUsed(indString material_name)
```

Parameters

`material_name`

Null-terminated ASCII string containing the name of the material.

Return Values

INDIGO_SUCCESS: Call was successful.

INDIGO_ERROR_GENERAL: A mesh is not currently being defined.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::addVertex

Adds a triangle vertex to the mesh.

```
indResult addVertex(const indVec3f pos, const indVec3f normal, const indVec2f texcoord_sets,
indUInt32 num_texcoord_sets)
```

Parameters

`pos`

position of the vertex in object coordinates.

`normal`

normal of the vertex in object coordinates. Only used if `normal_smoothing` is true for this mesh.

Vector should already be normalised.

`texcoord_sets`

Pointer to an array of *num_texcoord_sets* floating point pairs.

i.e. This pointer must be to an array of at least *num_texcoord_sets* * 2 floating point numbers.

`num_texcoord_sets`

Number of texture coordinate pairs passed in as the previous argument.

Return Values

INDIGO_SUCCESS: Call was successful.

INDIGO_ERROR_GENERAL: A mesh is not currently being defined.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::addTriangle

Adds a triangle vertex to the mesh.

```
indResult addTriangle(const indUInt32* vertex_indices, indUInt32 material_index)
```

Parameters

`vertex_indices`

0-based indices of the vertices making up the triangle. The indexed vertices must have already been defined. *vertex_indices* Should point to an array of 3 unsigned 32-bit integers.

`material_index`

Index of the material – the index into the list of materials added by calling `addMaterialUsed()`.

Return Values

INDIGO_SUCCESS: Call was successful.

INDIGO_ERROR_VERT_INDEX_OUT_OF_BOUNDS: One of the vertex indices was \geq the number of vertices already added to the mesh.

INDIGO_ERROR_GENERAL: A mesh is not currently being defined.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::addUVSetExposition

Exposes a specified texture coordinate (uv) set to the scene materials, as a named uv-set with a given name. This is necessary because materials specify the uv-set to use based on their name.

```
indResult addUVSetExposition(const indString uv_set_name, indUInt32 uv_set_index)
```

Parameters

`uv_set_name`

Null-terminated ASCII string containing the name that the uv-set will be exposed as.

`uv_set_index`

0-based Index of the texture coordinate (uv) set in the mesh. For example, the first set of uv-coordinates will have index 0.

Return Values

INDIGO_SUCCESS: Call was successful.

INDIGO_ERROR_GENERAL: A mesh is not currently being defined.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::finishMeshDefinition

Finish defining a mesh. Once this is called, no more operations can be done on the mesh – no more vertices can be added etc...

Once this call is made, the mesh is available to be instantiated in the scene by defining *models*.

```
indResult finishMeshDefinition()
```

Parameters

Return Values

INDIGO_SUCCESS: Call was successful.

INDIGO_ERROR_GENERAL: A mesh is not currently being defined.

Extra information

Header file: IndigoInterface1.h

IndigoInterface1::getLastErrorDescription

When some errors are encountered, extra information is available by calling this method, which returns an instance of the IndigoStringInterface1 interface.

```
IndigoStringInterface1* getLastErrorDescription()
```

Parameters

Return Values

A pointer to an instance of the IndigoStringInterface1 interface, encapsulating the error description string. This pointer should be assigned to an object of type *IndigoHandle<IndigoStringInterface1>*.

If there is no extra error description currently available, the empty string will be returned.

Extra information

Header file: IndigoInterface1.h

IndigoStringInterface1

Instances of this interface encapsulate ASCII strings.

IndigoStringInterface1::getString

Makes the string available as a const indString, e.g. As a const char*.

```
const indString getString()
```

Parameters

Return Values

Returns a pointer to the internal string.

Please note that this buffer should be used, or copied to another buffer, before the IndigoStringInterface1 object is destroyed.

Extra information

Header file: IndigoStringInterface1.h