# OptiX Utility Library

3.0.0

Generated by Doxygen 1.7.6.1

Wed Nov 21 2012 12:59:03

# Contents

# 1 Module Documentation

## 1.1 rtuTraversal: traversal API allowing batch raycasting queries utilizing either OptiX or the CPU.

### 1.1.1 Detailed Description

The OptiX traversal API is demonstrated in the traversal sample within the OptiX SDK.

#### Files

- file optixu_traversal.h

#### Typedefs

- typedef struct RTUtraversal_api ∗ RTUtraversal

#### Classes

- struct RTUtraversalresult

  *Structure encapsulating the result of a single ray query.*

#### Enumerations

- enum RTUquerytype { RTU_QUERY_TYPE_ANY_HIT = 0, RTU_QUERY_TYPE_CLOSEST_HIT, RTU_QUERY_TYPE_COUNT }
- enum RTUrayformat { RTU_RAYFORMAT_ORIGIN_DIRECTION_TMIN_TMAX_INTERLEAVED = 0, RTU_RAYFORMAT_ORIGIN_DIRECTION_INTERLEAVED, RTU_RAYFORMAT_COUNT }
- enum RTUtriformat { RTU_TRIFORMAT_MESH = 0, RTU_TRIFORMAT_TRIANGLE_SOUP, RTU_TRIFORMAT_COUNT }
- enum RTUinitoptions { RTU_INITOPTION_NONE = 0, RTU_INITOPTION_GPU_ONLY = 1 << 0, RTU_INITOPTION_CPU_ONLY = 1 << 1, RTU_INITOPTION_CULL_BACKFACE = 1 << 2 }
- enum RTUoutput { RTU_OUTPUT_NONE = 0, RTU_OUTPUT_NORMAL = 1 << 0, RTU_OUTPUT_BARYCENTRIC = 1 << 1, RTU_OUTPUT_BACKFACING = 1 << 2 }
- enum RTUoption { RTU_OPTION_INT_NUM_THREADS = 0 }

#### Functions

- RTresult RTAPI rtuTraversalCreate (RTUtraversal ∗traversal, RTUquerytype query_type, RTUrayformat ray_format, RTUtriformat tri_format, unsigned int outputs, unsigned int options, RTcontext context)
- RTresult RTAPI rtuTraversalGetErrorString (RTUtraversal traversal, RTresult code, const char ∗∗return_string)
- RTresult RTAPI rtuTraversalSetOption (RTUtraversal traversal, RTUoption option, void ∗value)
- RTresult RTAPI rtuTraversalSetMesh (RTUtraversal traversal, unsigned int num-_verts, const float ∗verts, unsigned int num_tris, const unsigned ∗indices)
- RTresult RTAPI rtuTraversalSetTriangles (RTUtraversal traversal, unsigned int num_tris, const float ∗tris)
- RTresult RTAPI rtuTraversalSetAccelData (RTUtraversal traversal, const void ∗data, RTsize data_size)
- RTresult RTAPI rtuTraversalGetAccelDataSize (RTUtraversal traversal, RTsize ∗data_size)
- RTresult RTAPI rtuTraversalGetAccelData (RTUtraversal traversal, void ∗data)
- RTresult RTAPI rtuTraversalMapRays (RTUtraversal traversal, unsigned int num-_rays, float ∗∗rays)
- RTresult RTAPI rtuTraversalUnmapRays (RTUtraversal traversal)
- RTresult RTAPI rtuTraversalPreprocess (RTUtraversal traversal)
- RTresult RTAPI rtuTraversalTraverse (RTUtraversal traversal)
- RTresult RTAPI rtuTraversalMapResults (RTUtraversal traversal, RTUtraversalresult ∗∗results)
- RTresult RTAPI rtuTraversalUnmapResults (RTUtraversal traversal)
- RTresult RTAPI rtuTraversalMapOutput (RTUtraversal traversal, RTUoutput which, void ∗∗output)
- RTresult RTAPI rtuTraversalUnmapOutput (RTUtraversal traversal, RTUoutput which)
- RTresult RTAPI rtuTraversalDestroy (RTUtraversal traversal)

### 1.1.2 Typedef Documentation

#### 1.1.2.1 typedef struct RTUtraversal_api∗ RTUtraversal

Opaque type. Note that the ∗_api types should never be used directly. Only the typedef target names will be guaranteed to remain unchanged.

Definition at line 116 of file optixu_traversal.h.

### 1.1.3 Enumeration Type Documentation

#### 1.1.3.1 enum RTUinitoptions

Initialization options (static across life of traversal object).

The rtuTraverse API supports both running on the CPU and GPU. When RTU_INIT-OPTION_NONE is specified GPU context creation is attempted. If that fails (such as when there isn't an NVIDIA GPU part present, the CPU code path is automatically chosen. Specifying RTU_INITOPTION_GPU_ONLY or RTU_INITOPTION_CPU_ONLY will only use the GPU or CPU modes without automatic transitions from one to the other.

RTU_INITOPTION_CULL_BACKFACE will enable back face culling during intersection.

**Enumerator:**

***RTU_INITOPTION_NONE***

***RTU_INITOPTION_GPU_ONLY***

***RTU_INITOPTION_CPU_ONLY***

***RTU_INITOPTION_CULL_BACKFACE***

Definition at line 89 of file optixu_traversal.h.

#### 1.1.3.2 enum RTUoption

Runtime options (can be set multiple times for a given traversal object).

**Enumerator:**

***RTU_OPTION_INT_NUM_THREADS***

Definition at line 107 of file optixu_traversal.h.

#### 1.1.3.3 enum RTUoutput

**Enumerator:**

***RTU_OUTPUT_NONE***

***RTU_OUTPUT_NORMAL***

***RTU_OUTPUT_BARYCENTRIC***

***RTU_OUTPUT_BACKFACING***

Definition at line 96 of file optixu_traversal.h.

#### 1.1.3.4 enum RTUquerytype

The type of ray query to be performed.

See OptiX Programming Guide for explanation of any vs. closest hit queries. Note that in the case of RTU_QUERY_TYPE_ANY_HIT, the prim_id and t intersection values in RTUtraversalresult will correspond to the first successful intersection. These values may not be indicative of the closest intersection, only that there was at least one.

**Enumerator:**

| | |
|---|---|
| ***RTU_QUERY_TYPE_ANY_HIT*** | Perform any hit calculation |
| ***RTU_QUERY_TYPE_CLOSEST_HIT*** | Perform closest hit calculation |
| ***RTU_QUERY_TYPE_COUNT*** | |

Definition at line 49 of file optixu_traversal.h.

#### 1.1.3.5 enum RTUrayformat

The input format of the ray vector.

**Enumerator:**

> ***RTU_RAYFORMAT_ORIGIN_DIRECTION_TMIN_TMAX_INTERLEAVED***
>
> ***RTU_RAYFORMAT_ORIGIN_DIRECTION_INTERLEAVED***
>
> ***RTU_RAYFORMAT_COUNT***

Definition at line 58 of file optixu_traversal.h.

#### 1.1.3.6 enum RTUtriformat

The input format of the triangles.

TRIANGLE_SOUP implies future use of rtuTraversalSetTriangles while MESH implies use of rtuTraversalSetMesh.

**Enumerator:**

> ***RTU_TRIFORMAT_MESH***
>
> ***RTU_TRIFORMAT_TRIANGLE_SOUP***
>
> ***RTU_TRIFORMAT_COUNT***

Definition at line 70 of file optixu_traversal.h.

### 1.1.4 Function Documentation

#### 1.1.4.1 RTresult RTAPI **rtuTraversalCreate** ( **RTUtraversal** ∗ *traversal,* **RTUquerytype** *query_type,* **RTUrayformat** *ray_format,* **RTUtriformat** *tri_format,* unsigned int *outputs,* unsigned int *options,* RTcontext *context* )

Create a traversal state and associate a context with it. If context is a null pointer a new context will be created internally. The context should also not be used for any other launch commands from the OptiX host API, nor attached to multiple RTUtraversal objects at one time.

**Parameters**

| | | |
|---|---|---|
| out | *traversal* | Return pointer for traverse state handle |
| | *query_type* | Ray query type |
| | *ray_format* | Ray format |
| | *tri_format* | Triangle format |
| | *outputs* | OR'ed mask of requested RTUoutputs |
| | *options* | Bit vector of or'ed RTUinitoptions. |
| | *context* | RTcontext used for internal object creation |

#### 1.1.4.2 RTresult RTAPI **rtuTraversalDestroy** ( **RTUtraversal** *traversal* )

Clean up any internal memory associated with rtuTraversal operations. Includes destruction of result buffers returned via rtuTraversalGetResults. Invalidates traversal object.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |

#### 1.1.4.3 RTresult RTAPI **rtuTraversalGetAccelData** ( **RTUtraversal** *traversal,* void ∗ *data* )

Retrieve acceleration data for current geometry. Will force acceleration build if necessary. The data parameter should be preallocated and its length should match return value of rtuTraversalGetAccelDataSize.

**Parameters**

| | | |
|---|---|---|
| | *traversal* | Traversal state handle |
| `out` | *data* | Acceleration data |

#### 1.1.4.4 RTresult RTAPI **rtuTraversalGetAccelDataSize** ( **RTUtraversal** *traversal,* RTsize ∗ *data_size* )

Retrieve acceleration data size for current geometry. Will force acceleration build if necessary.

**Parameters**

| | | |
|---|---|---|
| | *traversal* | Traversal state handle |
| `out` | *data_size* | Size of acceleration data |

#### 1.1.4.5 RTresult RTAPI **rtuTraversalGetErrorString** ( **RTUtraversal** *traversal,* RTresult *code,* const char ∗∗ *return_string* )

Returns the string associated with the error code and any additional information from the last error. If traversal is non-NULL return_string only remains valid while traversal is live.

**Parameters**

| | | |
|---|---|---|
| | *traversal* | Traversal state handle. Can be NULL. |
| | *code* | Error code from last error |
| `out` | *return_string* | Pointer to string with error message in it. |

#### 1.1.4.6 RTresult RTAPI **rtuTraversalMapOutput** ( **RTUtraversal** *traversal,* **RTUoutput** *which,* void ∗∗ *output* )

Retrieve user-specified output from last rtuTraversal call. Output can be copied from the pointer returned by rtuTraversalMapOutput and will have length 'num_rays' from as prescribed from the previous call to rtuTraversalSetRays. For each RTUoutput, a single rtuTraversalMapOutput pointers can be outstanding. rtuTraversalUnmapOutput should be called when finished reading the output.

If requested output type was not turned on with a previous call to rtuTraverseSetOutputs an error will be returned. See RTUoutput enum for description of output data formats for various outputs.

**Parameters**

| | | |
|---|---|---|
| | *traversal* | Traversal state handle |
| | *which* | Output type to be specified |
| `out` | *output* | Pointer to output from last traverse |

#### 1.1.4.7 RTresult RTAPI rtuTraversalMapRays ( RTUtraversal *traversal,* unsigned int *num_rays,* float ∗∗ *rays* )

Specify set of rays to be cast upon next call to rtuTraversalTraverse. rtuTraversalMapRays obtains a pointer which can be used to copy the ray data into. Rays should be packed in the format described in rtuTraversalCreate call. When copying is completed rtuTraversalUnmapRays should be called. Note that this call invalidates any existing results buffers until rtuTraversalTraverse is called again.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |
| *num_rays* | Number of rays to be traced |
| *rays* | Pointer to ray data |

#### 1.1.4.8 RTresult RTAPI rtuTraversalMapResults ( RTUtraversal *traversal,* RTUtraversalresult ∗∗ *results* )

Retrieve results of last rtuTraversal call. Results can be copied from the pointer returned by rtuTraversalMapResults and will have length 'num_rays' as prescribed from the previous call to rtuTraversalMapRays. rtuTraversalUnmapResults should be called when finished reading the results. Returned primitive ID of -1 indicates a ray miss.

**Parameters**

| | | |
|---|---|---|
| | *traversal* | Traversal state handle |
| `out` | *results* | Pointer to results of last traverse |

#### 1.1.4.9 RTresult RTAPI rtuTraversalPreprocess ( RTUtraversal *traversal* )

Perform any necessary preprocessing (eg, acceleration structure building, optix context compilation). It is not necessary to call this function as rtuTraversalTraverse will call this internally as necessary.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |

#### 1.1.4.10 RTresult RTAPI **rtuTraversalSetAccelData** ( **RTUtraversal** *traversal,* const void ∗ *data,* RTsize *data_size* )

Specify acceleration data for current geometry. Input acceleration data should be result of rtuTraversalGetAccelData or rtAccelerationGetData call.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |
| *data* | Acceleration data |
| *data_size* | Size of acceleration data |

#### 1.1.4.11 RTresult RTAPI **rtuTraversalSetMesh** ( **RTUtraversal** *traversal,* unsigned int *num_verts,* const float ∗ *verts,* unsigned int *num_tris,* const unsigned ∗ *indices* )

Specify triangle mesh to be intersected by the next call to rtuTraversalLaunch. Only one geometry set may be active at a time. Subsequent calls to rtuTraversalSetTriangles or rtuTraversalSetMesh will override any previously specified geometry. No internal copies of the mesh data are made. The user should ensure that the mesh data remains valid until after rtuTraversalTraverse has been called. Counter-clockwise winding is assumed for normal and backfacing computations.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |
| *num_verts* | Vertex count |
| *verts* | Vertices [ v1_x, v1_y, v1_z, v2.x, ... ] |
| *num_tris* | Triangle count |
| *indices* | Indices [ tri1_index1, tr1_index2, ... ] |

#### 1.1.4.12 RTresult RTAPI **rtuTraversalSetOption** ( **RTUtraversal** *traversal,* **RTUoption** *option,* void ∗ *value* )

Set a runtime option. Unlike initialization options, these options may be set more than once for a given RTUtraversal instance.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |
| *option* | The option to be set |
| *value* | Value of the option |

#### 1.1.4.13 RTresult RTAPI **rtuTraversalSetTriangles** ( **RTUtraversal** *traversal,* unsigned int *num_tris,* const float ∗ *tris* )

Specify triangle soup to be intersected by the next call to rtuTraversalLaunch. Only one geometry set may be active at a time. Subsequent calls to rtuTraversalSetTriangles or rtuTraversalSetMesh will override any previously specified geometry. No internal copies of the triangle data are made. The user should ensure that the triangle data remains valid until after rtuTraversalTraverse has been called. Counter-clockwise winding is assumed for normal and backfacing computations.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |
| *num_tris* | Triangle count |
| *tris* | Triangles [ tri1_v1.x, tri1_v1.y, tr1_v1.z, tri1_v2.x, ... ] |

#### 1.1.4.14 RTresult RTAPI **rtuTraversalTraverse** ( **RTUtraversal** *traversal* )

Perform any necessary preprocessing (eg, acceleration structure building and kernel compilation ) and cast current rays against current geometry.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |

#### 1.1.4.15 RTresult RTAPI **rtuTraversalUnmapOutput** ( **RTUtraversal** *traversal,* **RTUoutput** *which* )

See rtuTraversalMapOutput

#### 1.1.4.16 RTresult RTAPI **rtuTraversalUnmapRays** ( **RTUtraversal** *traversal* )

See rtuTraversalMapRays.

#### 1.1.4.17 RTresult RTAPI **rtuTraversalUnmapResults** ( **RTUtraversal** *traversal* )

See rtuTraversalMapResults

## 1.2 OptiXpp: C++ wrapper for the OptiX C API.

### 1.2.1 Detailed Description

OptiXpp wraps each OptiX C API opaque type in a C++ class. Most of the OptiXpp class member functions map directly to C API function calls:

- VariableObj::getContext -> rtVariableGetContext
- ContextObj::createBuffer -> rtBufferCreate

Many classes have convenience functions which encapsulate a related group of OptiX functions. For instance

```
ContextObj::createBuffer(unsigned int type, RTformat format, RTsize width)
```

provides the functionality of

- `rtBufferCreate`
- `rtBufferSetFormat`
- `rtBufferSetSize1D`

in a single call.

Manipulation of these classes is performed via reference counted Handle class. Rather than working with a ContextObj directly you would use a Context instead, which is simply a typedef for *Handle<ContextObj>*. The OptiX SDK has many examples of the use of OptiXpp. In particular, sample5 and sample5pp are a good place to look when learning OptiXpp as they are nearly identical programs, one created with the C API and one with the C++ API.

#### Files

- file optixpp_namespace.h

#### Typedefs

- typedef Handle< AccelerationObj > optix::Acceleration
- typedef Handle< BufferObj > optix::Buffer
- typedef Handle< ContextObj > optix::Context
- typedef Handle< GeometryObj > optix::Geometry
- typedef Handle< GeometryGroupObj > optix::GeometryGroup
- typedef Handle < GeometryInstanceObj > optix::GeometryInstance
- typedef Handle< GroupObj > optix::Group
- typedef Handle< MaterialObj > optix::Material
- typedef Handle< ProgramObj > optix::Program
- typedef Handle< SelectorObj > optix::Selector
- typedef Handle< TextureSamplerObj > optix::TextureSampler
- typedef Handle< TransformObj > optix::Transform
- typedef Handle< VariableObj > optix::Variable

### Classes

- class optix::Handle< T >

    *The Handle class is a reference counted handle class used to manipulate API objects.*
- class optix::Exception

    *Exception class for error reporting from the OptiXpp API.*
- class optix::APIObj

    *Base class for all reference counted wrappers around OptiX C API opaque types.*
- class optix::DestroyableObj

    *Base class for all wrapper objects which can be destroyed and validated.*
- class optix::ScopedObj

    *Base class for all objects which are OptiX variable containers.*
- class optix::VariableObj

    *Variable object wraps OptiX C API RTvariable type and its related function set.*
- class optix::ContextObj

    *Context object wraps the OptiX C API RTcontext opaque type and its associated function set.*
- class optix::ProgramObj

    *Program object wraps the OptiX C API RTprogram opaque type and its associated function set.*
- class optix::GroupObj

    *Group wraps the OptiX C API RTgroup opaque type and its associated function set.*
- class optix::GeometryGroupObj

    *GeometryGroup wraps the OptiX C API RTgeometrygroup opaque type and its associated function set.*
- class optix::TransformObj

    *Transform wraps the OptiX C API RTtransform opaque type and its associated function set.*
- class optix::SelectorObj

    *Selector wraps the OptiX C API RTselector opaque type and its associated function set.*
- class optix::AccelerationObj

    *Acceleration wraps the OptiX C API RTacceleration opaque type and its associated function set.*
- class optix::GeometryInstanceObj

    *GeometryInstance wraps the OptiX C API RTgeometryinstance acceleration opaque type and its associated function set.*
- class optix::GeometryObj

    *Geometry wraps the OptiX C API RTgeometry opaque type and its associated function set.*
- class optix::MaterialObj

    *Material wraps the OptiX C API RTmaterial opaque type and its associated function set.*
- class optix::TextureSamplerObj

    *TextureSampler wraps the OptiX C API RTtexturesampler opaque type and its associated function set.*

- class optix::BufferObj

  *Buffer wraps the OptiX C API RTbuffer opaque type and its associated function set.*

### Functions

- static Exception optix::Exception::makeException (RTresult code, RTcontext context)
- static Exception optix::APIObj::makeException (RTresult code, RTcontext context)
- Handle$<$ VariableObj $>$ optix::Handle$<$ T $>$::operator[] (const std::string &varname)
- Handle$<$ VariableObj $>$ optix::Handle$<$ T $>$::operator[] (const char $*$varname)
- virtual void optix::APIObj::checkError (RTresult code) const
- virtual void optix::APIObj::checkError (RTresult code, Context context) const
- void optix::APIObj::checkErrorNoGetContext (RTresult code) const
- Context optix::ContextObj::getContext () const
- static unsigned int optix::ContextObj::getDeviceCount ()
- static std::string optix::ContextObj::getDeviceName (int ordinal)
- static void optix::ContextObj::getDeviceAttribute (int ordinal, RTdeviceattribute attrib, RTsize size, void $*$p)
- static Context optix::ContextObj::create ()
- void optix::ContextObj::destroy ()
- void optix::ContextObj::validate ()
- void optix::ContextObj::compile ()
- int optix::ContextObj::getRunningState () const
- RTcontext optix::ContextObj::get ()
- void optix::ProgramObj::destroy ()
- void optix::ProgramObj::validate ()
- Context optix::ProgramObj::getContext () const
- Variable optix::ProgramObj::declareVariable (const std::string &name)
- Variable optix::ProgramObj::queryVariable (const std::string &name) const
- void optix::ProgramObj::removeVariable (Variable v)
- unsigned int optix::ProgramObj::getVariableCount () const
- Variable optix::ProgramObj::getVariable (unsigned int index) const
- RTprogram optix::ProgramObj::get ()
- void optix::GroupObj::destroy ()
- void optix::GroupObj::validate ()
- Context optix::GroupObj::getContext () const
- void optix::SelectorObj::destroy ()
- void optix::SelectorObj::validate ()
- Context optix::SelectorObj::getContext () const
- RTselector optix::SelectorObj::get ()
- RTgroup optix::GroupObj::get ()
- void optix::GeometryGroupObj::destroy ()
- void optix::GeometryGroupObj::validate ()
- Context optix::GeometryGroupObj::getContext () const

- RTgeometrygroup optix::GeometryGroupObj::get ()
- void optix::TransformObj::destroy ()
- void optix::TransformObj::validate ()
- Context optix::TransformObj::getContext () const
- RTtransform optix::TransformObj::get ()
- void optix::AccelerationObj::destroy ()
- void optix::AccelerationObj::validate ()
- Context optix::AccelerationObj::getContext () const
- RTacceleration optix::AccelerationObj::get ()
- void optix::GeometryInstanceObj::destroy ()
- void optix::GeometryInstanceObj::validate ()
- Context optix::GeometryInstanceObj::getContext () const
- RTgeometryinstance optix::GeometryInstanceObj::get ()
- void optix::GeometryObj::destroy ()
- void optix::GeometryObj::validate ()
- Context optix::GeometryObj::getContext () const
- RTgeometry optix::GeometryObj::get ()
- void optix::MaterialObj::destroy ()
- void optix::MaterialObj::validate ()
- Context optix::MaterialObj::getContext () const
- RTmaterial optix::MaterialObj::get ()
- void optix::TextureSamplerObj::destroy ()
- void optix::TextureSamplerObj::validate ()
- Context optix::TextureSamplerObj::getContext () const
- RTtexturesampler optix::TextureSamplerObj::get ()
- void optix::BufferObj::destroy ()
- void optix::BufferObj::validate ()
- Context optix::BufferObj::getContext () const
- RTbuffer optix::BufferObj::get ()
- Context optix::VariableObj::getContext () const
- std::string optix::VariableObj::getName () const
- std::string optix::VariableObj::getAnnotation () const
- RTobjecttype optix::VariableObj::getType () const
- RTvariable optix::VariableObj::get ()
- RTsize optix::VariableObj::getSize () const

#### Float setters

Set variable to have a float value.

- void optix::VariableObj::setFloat (float f1)
- void optix::VariableObj::setFloat (optix::float2 f)
- void optix::VariableObj::setFloat (float f1, float f2)
- void optix::VariableObj::setFloat (optix::float3 f)
- void optix::VariableObj::setFloat (float f1, float f2, float f3)
- void optix::VariableObj::setFloat (optix::float4 f)

- void optix::VariableObj::setFloat (float f1, float f2, float f3, float f4)
- void optix::VariableObj::set1fv (const float ∗f)
- void optix::VariableObj::set2fv (const float ∗f)
- void optix::VariableObj::set3fv (const float ∗f)
- void optix::VariableObj::set4fv (const float ∗f)

### Int setters

Set variable to have an int value.

- void optix::VariableObj::setInt (int i1)
- void optix::VariableObj::setInt (optix::int2 i)
- void optix::VariableObj::setInt (int i1, int i2)
- void optix::VariableObj::setInt (optix::int3 i)
- void optix::VariableObj::setInt (int i1, int i2, int i3)
- void optix::VariableObj::setInt (optix::int4 i)
- void optix::VariableObj::setInt (int i1, int i2, int i3, int i4)
- void optix::VariableObj::set1iv (const int ∗i)
- void optix::VariableObj::set2iv (const int ∗i)
- void optix::VariableObj::set3iv (const int ∗i)
- void optix::VariableObj::set4iv (const int ∗i)

### Unsigned int setters

Set variable to have an unsigned int value.

- void optix::VariableObj::setUint (unsigned int u1)
- void optix::VariableObj::setUint (unsigned int u1, unsigned int u2)
- void optix::VariableObj::setUint (unsigned int u1, unsigned int u2, unsigned int u3)
- void optix::VariableObj::setUint (unsigned int u1, unsigned int u2, unsigned int u3, unsigned int u4)
- void optix::VariableObj::setUint (optix::uint2 u)
- void optix::VariableObj::setUint (optix::uint3 u)
- void optix::VariableObj::setUint (optix::uint4 u)
- void optix::VariableObj::set1uiv (const unsigned int ∗u)
- void optix::VariableObj::set2uiv (const unsigned int ∗u)
- void optix::VariableObj::set3uiv (const unsigned int ∗u)
- void optix::VariableObj::set4uiv (const unsigned int ∗u)

### Matrix setters

Set variable to have a Matrix value

- void optix::VariableObj::setMatrix2x2fv (bool transpose, const float ∗m)
- void optix::VariableObj::setMatrix2x3fv (bool transpose, const float ∗m)

- void optix::VariableObj::setMatrix2x4fv (bool transpose, const float ∗m)
- void optix::VariableObj::setMatrix3x2fv (bool transpose, const float ∗m)
- void optix::VariableObj::setMatrix3x3fv (bool transpose, const float ∗m)
- void optix::VariableObj::setMatrix3x4fv (bool transpose, const float ∗m)
- void optix::VariableObj::setMatrix4x2fv (bool transpose, const float ∗m)
- void optix::VariableObj::setMatrix4x3fv (bool transpose, const float ∗m)
- void optix::VariableObj::setMatrix4x4fv (bool transpose, const float ∗m)

#### Numeric value getters

Query value of a variable with scalar numeric value

- float optix::VariableObj::getFloat () const
- unsigned int optix::VariableObj::getUint () const
- int optix::VariableObj::getInt () const

#### OptiX API object setters

Set variable to have an OptiX API object as its value

- void optix::VariableObj::setBuffer (Buffer buffer)
- void optix::VariableObj::set (Buffer buffer)
- void optix::VariableObj::setTextureSampler (TextureSampler texturesample)

#### OptiX API object getters

Reitrieve OptiX API object value from a variable

- Buffer optix::VariableObj::getBuffer () const
- TextureSampler optix::VariableObj::getTextureSampler () const
- Program optix::VariableObj::getProgram () const

#### User data variable accessors

- void optix::VariableObj::setUserData (RTsize size, const void ∗ptr)
- void optix::VariableObj::getUserData (RTsize size, void ∗ptr) const

- void optix::ContextObj::checkError (RTresult code) const
- std::string optix::ContextObj::getErrorString (RTresult code) const

- Acceleration optix::ContextObj::createAcceleration (const char ∗builder, const char ∗traverser)
- Buffer optix::ContextObj::createBuffer (unsigned int type)
- Buffer optix::ContextObj::createBuffer (unsigned int type, RTformat format)

- Buffer optix::ContextObj::createBuffer (unsigned int type, RTformat format, RTsize width)
- Buffer optix::ContextObj::createBuffer (unsigned int type, RTformat format, RTsize width, RTsize height)
- Buffer optix::ContextObj::createBuffer (unsigned int type, RTformat format, RTsize width, RTsize height, RTsize depth)
- Buffer optix::ContextObj::createBufferForCUDA (unsigned int type)
- Buffer optix::ContextObj::createBufferForCUDA (unsigned int type, RTformat format)
- Buffer optix::ContextObj::createBufferForCUDA (unsigned int type, RTformat format, RTsize width)
- Buffer optix::ContextObj::createBufferForCUDA (unsigned int type, RTformat format, RTsize width, RTsize height)
- Buffer optix::ContextObj::createBufferForCUDA (unsigned int type, RTformat format, RTsize width, RTsize height, RTsize depth)
- Buffer optix::ContextObj::createBufferFromGLBO (unsigned int type, unsigned int vbo)
- TextureSampler optix::ContextObj::createTextureSamplerFromGLImage (unsigned int id, RTgltarget target)
- Geometry optix::ContextObj::createGeometry ()
- GeometryInstance optix::ContextObj::createGeometryInstance ()
- template<class Iterator >
  GeometryInstance optix::ContextObj::createGeometryInstance (Geometry geometry, Iterator matlbegin, Iterator matlend)
- Group optix::ContextObj::createGroup ()
- template<class Iterator >
  Group optix::ContextObj::createGroup (Iterator childbegin, Iterator childend)
- GeometryGroup optix::ContextObj::createGeometryGroup ()
- template<class Iterator >
  GeometryGroup optix::ContextObj::createGeometryGroup (Iterator childbegin, Iterator childend)
- Transform optix::ContextObj::createTransform ()
- Material optix::ContextObj::createMaterial ()
- Program optix::ContextObj::createProgramFromPTXFile (const std::string &ptx, const std::string &program_name)
- Program optix::ContextObj::createProgramFromPTXString (const std::string &ptx, const std::string &program_name)
- Selector optix::ContextObj::createSelector ()
- TextureSampler optix::ContextObj::createTextureSampler ()

- template<class Iterator >
  void optix::ContextObj::setDevices (Iterator begin, Iterator end)
- std::vector< int > optix::ContextObj::getEnabledDevices () const
- unsigned int optix::ContextObj::getEnabledDeviceCount () const

- int optix::ContextObj::getMaxTextureCount () const
- int optix::ContextObj::getCPUNumThreads () const

- RTsize optix::ContextObj::getUsedHostMemory () const
- int optix::ContextObj::getGPUPagingActive () const
- int optix::ContextObj::getGPUPagingForcedOff () const
- RTsize optix::ContextObj::getAvailableDeviceMemory (int ordinal) const

- void optix::ContextObj::setCPUNumThreads (int cpu_num_threads)
- void optix::ContextObj::setGPUPagingForcedOff (int gpu_paging_forced_off)

- void optix::ContextObj::setStackSize (RTsize stack_size_bytes)
- RTsize optix::ContextObj::getStackSize () const
- void optix::ContextObj::setTimeoutCallback (RTtimeoutcallback callback, double min_polling_seconds)
- void optix::ContextObj::setEntryPointCount (unsigned int num_entry_points)
- unsigned int optix::ContextObj::getEntryPointCount () const
- void optix::ContextObj::setRayTypeCount (unsigned int num_ray_types)
- unsigned int optix::ContextObj::getRayTypeCount () const

- void optix::ContextObj::setRayGenerationProgram (unsigned int entry_point_-index, Program program)
- Program optix::ContextObj::getRayGenerationProgram (unsigned int entry_point-_index) const
- void optix::ContextObj::setExceptionProgram (unsigned int entry_point_index, -Program program)
- Program optix::ContextObj::getExceptionProgram (unsigned int entry_point_-index) const
- void optix::ContextObj::setExceptionEnabled (RTexception exception, bool en-abled)
- bool optix::ContextObj::getExceptionEnabled (RTexception exception) const
- void optix::ContextObj::setMissProgram (unsigned int ray_type_index, Program program)
- Program optix::ContextObj::getMissProgram (unsigned int ray_type_index) const

- void optix::ContextObj::launch (unsigned int entry_point_index, RTsize image_-width)
- void optix::ContextObj::launch (unsigned int entry_point_index, RTsize image_-width, RTsize image_height)
- void optix::ContextObj::launch (unsigned int entry_point_index, RTsize image_-width, RTsize image_height, RTsize image_depth)

- void optix::ContextObj::setPrintEnabled (bool enabled)
- bool optix::ContextObj::getPrintEnabled () const
- void optix::ContextObj::setPrintBufferSize (RTsize buffer_size_bytes)
- RTsize optix::ContextObj::getPrintBufferSize () const
- void optix::ContextObj::setPrintLaunchIndex (int x, int y=-1, int z=-1)
- optix::int3 optix::ContextObj::getPrintLaunchIndex () const

- Variable optix::ContextObj::declareVariable (const std::string &name)

- Variable optix::ContextObj::queryVariable (const std::string &name) const
- void optix::ContextObj::removeVariable (Variable v)
- unsigned int optix::ContextObj::getVariableCount () const
- Variable optix::ContextObj::getVariable (unsigned int index) const

- void optix::GroupObj::setAcceleration (Acceleration acceleration)
- Acceleration optix::GroupObj::getAcceleration () const

- void optix::GroupObj::setChildCount (unsigned int count)
- unsigned int optix::GroupObj::getChildCount () const
- template<typename T >
  void optix::GroupObj::setChild (unsigned int index, T child)
- template<typename T >
  T optix::GroupObj::getChild (unsigned int index) const

- void optix::GeometryGroupObj::setAcceleration (Acceleration acceleration)
- Acceleration optix::GeometryGroupObj::getAcceleration () const

- void optix::GeometryGroupObj::setChildCount (unsigned int count)
- unsigned int optix::GeometryGroupObj::getChildCount () const
- void optix::GeometryGroupObj::setChild (unsigned int index, GeometryInstance geometryinstance)
- GeometryInstance optix::GeometryGroupObj::getChild (unsigned int index) const

- template<typename T >
  void optix::TransformObj::setChild (T child)
- template<typename T >
  T optix::TransformObj::getChild () const

- void optix::TransformObj::setMatrix (bool transpose, const float ∗matrix, const float ∗inverse_matrix)
- void optix::TransformObj::getMatrix (bool transpose, float ∗matrix, float ∗inverse_matrix) const

- void optix::SelectorObj::setVisitProgram (Program program)
- Program optix::SelectorObj::getVisitProgram () const

- void optix::SelectorObj::setChildCount (unsigned int count)
- unsigned int optix::SelectorObj::getChildCount () const
- template<typename T >
  void optix::SelectorObj::setChild (unsigned int index, T child)
- template<typename T >
  T optix::SelectorObj::getChild (unsigned int index) const

- Variable optix::SelectorObj::declareVariable (const std::string &name)
- Variable optix::SelectorObj::queryVariable (const std::string &name) const

- void optix::SelectorObj::removeVariable (Variable v)
- unsigned int optix::SelectorObj::getVariableCount () const
- Variable optix::SelectorObj::getVariable (unsigned int index) const

- void optix::AccelerationObj::markDirty ()
- bool optix::AccelerationObj::isDirty () const

- void optix::AccelerationObj::setProperty (const std::string &name, const std::string &value)
- std::string optix::AccelerationObj::getProperty (const std::string &name) const
- void optix::AccelerationObj::setBuilder (const std::string &builder)
- std::string optix::AccelerationObj::getBuilder () const
- void optix::AccelerationObj::setTraverser (const std::string &traverser)
- std::string optix::AccelerationObj::getTraverser () const

- RTsize optix::AccelerationObj::getDataSize () const
- void optix::AccelerationObj::getData (void ∗data) const
- void optix::AccelerationObj::setData (const void ∗data, RTsize size)

- void optix::GeometryInstanceObj::setGeometry (Geometry geometry)
- Geometry optix::GeometryInstanceObj::getGeometry () const
- void optix::GeometryInstanceObj::setMaterialCount (unsigned int count)
- unsigned int optix::GeometryInstanceObj::getMaterialCount () const
- void optix::GeometryInstanceObj::setMaterial (unsigned int idx, Material material)
- Material optix::GeometryInstanceObj::getMaterial (unsigned int idx) const
- unsigned int optix::GeometryInstanceObj::addMaterial (Material material)

- Variable optix::GeometryInstanceObj::declareVariable (const std::string &name)
- Variable optix::GeometryInstanceObj::queryVariable (const std::string &name) const
- void optix::GeometryInstanceObj::removeVariable (Variable v)
- unsigned int optix::GeometryInstanceObj::getVariableCount () const
- Variable optix::GeometryInstanceObj::getVariable (unsigned int index) const

- void optix::GeometryObj::markDirty ()
- bool optix::GeometryObj::isDirty () const

- void optix::GeometryObj::setPrimitiveCount (unsigned int num_primitives)
- unsigned int optix::GeometryObj::getPrimitiveCount () const

- void optix::GeometryObj::setBoundingBoxProgram (Program program)
- Program optix::GeometryObj::getBoundingBoxProgram () const
- void optix::GeometryObj::setIntersectionProgram (Program program)
- Program optix::GeometryObj::getIntersectionProgram () const

- Variable optix::GeometryObj::declareVariable (const std::string &name)

- Variable optix::GeometryObj::queryVariable (const std::string &name) const
- void optix::GeometryObj::removeVariable (Variable v)
- unsigned int optix::GeometryObj::getVariableCount () const
- Variable optix::GeometryObj::getVariable (unsigned int index) const

- void optix::MaterialObj::setClosestHitProgram (unsigned int ray_type_index, - Program program)
- Program optix::MaterialObj::getClosestHitProgram (unsigned int ray_type_index) const
- void optix::MaterialObj::setAnyHitProgram (unsigned int ray_type_index, - Program program)
- Program optix::MaterialObj::getAnyHitProgram (unsigned int ray_type_index) const

- Variable optix::MaterialObj::declareVariable (const std::string &name)
- Variable optix::MaterialObj::queryVariable (const std::string &name) const
- void optix::MaterialObj::removeVariable (Variable v)
- unsigned int optix::MaterialObj::getVariableCount () const
- Variable optix::MaterialObj::getVariable (unsigned int index) const

- void optix::TextureSamplerObj::setMipLevelCount (unsigned int num_mip_levels)
- unsigned int optix::TextureSamplerObj::getMipLevelCount () const
- void optix::TextureSamplerObj::setArraySize (unsigned int num_textures_in_-array)
- unsigned int optix::TextureSamplerObj::getArraySize () const
- void optix::TextureSamplerObj::setWrapMode (unsigned int dim, RTwrapmode wrapmode)
- RTwrapmode optix::TextureSamplerObj::getWrapMode (unsigned int dim) const
- void optix::TextureSamplerObj::setFilteringModes (RTfiltermode minification, R-Tfiltermode magnification, RTfiltermode mipmapping)
- void optix::TextureSamplerObj::getFilteringModes (RTfiltermode &minification, R-Tfiltermode &magnification, RTfiltermode &mipmapping) const
- void optix::TextureSamplerObj::setMaxAnisotropy (float value)
- float optix::TextureSamplerObj::getMaxAnisotropy () const
- void optix::TextureSamplerObj::setReadMode (RTtexturereadmode readmode)
- RTtexturereadmode optix::TextureSamplerObj::getReadMode () const
- void optix::TextureSamplerObj::setIndexingMode (RTtextureindexmode index-mode)
- RTtextureindexmode optix::TextureSamplerObj::getIndexingMode () const

- int optix::TextureSamplerObj::getId () const

- void optix::TextureSamplerObj::setBuffer (unsigned int texture_array_idx, un-signed int mip_level, Buffer buffer)
- Buffer optix::TextureSamplerObj::getBuffer (unsigned int texture_array_idx, un-signed int mip_level) const

- void optix::TextureSamplerObj::registerGLTexture ()
- void optix::TextureSamplerObj::unregisterGLTexture ()

- void optix::BufferObj::setFormat (RTformat format)
- RTformat optix::BufferObj::getFormat () const
- void optix::BufferObj::setElementSize (RTsize size_of_element)
- RTsize optix::BufferObj::getElementSize () const
- void optix::BufferObj::getDevicePointer (unsigned int optix_device_number, C-Udeviceptr ∗device_pointer)
- void optix::BufferObj::setDevicePointer (unsigned int optix_device_number, C-Udeviceptr device_pointer)
- void optix::BufferObj::markDirty ()
- void optix::BufferObj::setSize (RTsize width)
- void optix::BufferObj::getSize (RTsize &width) const
- void optix::BufferObj::setSize (RTsize width, RTsize height)
- void optix::BufferObj::getSize (RTsize &width, RTsize &height) const
- void optix::BufferObj::setSize (RTsize width, RTsize height, RTsize depth)
- void optix::BufferObj::getSize (RTsize &width, RTsize &height, RTsize &depth) const
- void optix::BufferObj::setSize (unsigned int dimensionality, const RTsize ∗dims)
- void optix::BufferObj::getSize (unsigned int dimensionality, RTsize ∗dims) const
- unsigned int optix::BufferObj::getDimensionality () const

- unsigned int optix::BufferObj::getGLBOId () const
- void optix::BufferObj::registerGLBuffer ()
- void optix::BufferObj::unregisterGLBuffer ()

- void ∗ optix::BufferObj::map ()
- void optix::BufferObj::unmap ()

### 1.2.2 Typedef Documentation

#### 1.2.2.1 typedef Handle<AccelerationObj> optix::Acceleration

Use this to manipulate RTacceleration objects.

Definition at line 211 of file optixpp_namespace.h.

#### 1.2.2.2 typedef Handle<BufferObj> optix::Buffer

Use this to manipulate RTbuffer objects.

Definition at line 212 of file optixpp_namespace.h.

#### 1.2.2.3 typedef Handle<ContextObj> optix::Context

Use this to manipulate RTcontext objects.

Definition at line 213 of file optixpp_namespace.h.

#### 1.2.2.4 typedef Handle<GeometryObj> optix::Geometry

Use this to manipulate RTgeometry objects.

Definition at line 214 of file optixpp_namespace.h.

#### 1.2.2.5 typedef Handle<GeometryGroupObj> optix::GeometryGroup

Use this to manipulate RTgeometrygroup objects.

Definition at line 215 of file optixpp_namespace.h.

#### 1.2.2.6 typedef Handle<GeometryInstanceObj> optix::GeometryInstance

Use this to manipulate RTgeometryinstance objects.

Definition at line 216 of file optixpp_namespace.h.

#### 1.2.2.7 typedef Handle<GroupObj> optix::Group

Use this to manipulate RTgroup objects.

Definition at line 217 of file optixpp_namespace.h.

#### 1.2.2.8 typedef Handle<MaterialObj> optix::Material

Use this to manipulate RTmaterial objects.

Definition at line 218 of file optixpp_namespace.h.

#### 1.2.2.9 typedef Handle<ProgramObj> optix::Program

Use this to manipulate RTprogram objects.

Definition at line 219 of file optixpp_namespace.h.

#### 1.2.2.10 typedef Handle<SelectorObj> optix::Selector

Use this to manipulate RTselector objects.

Definition at line 220 of file optixpp_namespace.h.

#### 1.2.2.11 typedef Handle<TextureSamplerObj> optix::TextureSampler

Use this to manipulate RTtexturesampler objects.

Definition at line 221 of file optixpp_namespace.h.

#### 1.2.2.12 typedef Handle<TransformObj> optix::Transform

Use this to manipulate RTtransform objects.

Definition at line 222 of file optixpp_namespace.h.

#### 1.2.2.13 typedef Handle<VariableObj> optix::Variable

Use this to manipulate RTvariable objects.

Definition at line 223 of file optixpp_namespace.h.

### 1.2.3 Function Documentation

#### 1.2.3.1 unsigned int **optix::GeometryInstanceObj::addMaterial** ( **Material** *material* ) `[inline]`

Adds the provided material and returns the index to newly added material; increases material count by one.

Definition at line 2631 of file optixpp_namespace.h.

#### 1.2.3.2 void **optix::APIObj::checkError** ( **RTresult** *code* ) const `[inline, virtual]`

Check the given result code and throw an error with appropriate message if the code is not RTsuccess

Reimplemented in optix::ContextObj.

Definition at line 1523 of file optixpp_namespace.h.

#### 1.2.3.3 void **optix::APIObj::checkError** ( **RTresult** *code,* **Context** *context* ) const `[inline, virtual]`

Definition at line 1531 of file optixpp_namespace.h.

#### 1.2.3.4 void **optix::ContextObj::checkError** ( **RTresult** *code* ) const `[inline, virtual]`

See APIObj::checkError

Reimplemented from optix::APIObj.

Definition at line 1551 of file optixpp_namespace.h.

#### 1.2.3.5 void **optix::APIObj::checkErrorNoGetContext** ( **RTresult** *code* ) const `[inline]`

Definition at line 1539 of file optixpp_namespace.h.

#### 1.2.3.6 void **optix::ContextObj::compile** ( ) `[inline]`

See rtContextCompile.

Definition at line 2061 of file optixpp_namespace.h.

#### 1.2.3.7 Context **optix::ContextObj::create** ( ) `[inline, static]`

Creates a Context object. See rtContextCreate.

Definition at line 1582 of file optixpp_namespace.h.

#### 1.2.3.8 Acceleration **optix::ContextObj::createAcceleration** ( const char * *builder*, const char * *traverser* ) `[inline]`

See rtAccelerationCreate

Definition at line 1602 of file optixpp_namespace.h.

#### 1.2.3.9 Buffer **optix::ContextObj::createBuffer** ( unsigned int *type* ) `[inline]`

Create a buffer with given RTbuffertype. See rtBufferCreate.

Definition at line 1612 of file optixpp_namespace.h.

#### 1.2.3.10 Buffer **optix::ContextObj::createBuffer** ( unsigned int *type*, RTformat *format* ) `[inline]`

Create a buffer with given RTbuffertype and RTformat. See rtBufferCreate, rtBufferSetFormat.

Definition at line 1619 of file optixpp_namespace.h.

#### 1.2.3.11 Buffer **optix::ContextObj::createBuffer** ( unsigned int *type*, RTformat *format*, RTsize *width* ) `[inline]`

Create a buffer with given RTbuffertype, RTformat and dimension. See rtBufferCreate, rtBufferSetFormat and rtBufferSetSize1D.

Definition at line 1627 of file optixpp_namespace.h.

#### 1.2.3.12 Buffer **optix::ContextObj::createBuffer** ( unsigned int *type*, RTformat *format*, RTsize *width*, RTsize *height* ) `[inline]`

Create a buffer with given RTbuffertype, RTformat and dimension. See rtBufferCreate, rtBufferSetFormat and rtBufferSetSize2D.

Definition at line 1636 of file optixpp_namespace.h.

#### 1.2.3.13 Buffer **optix::ContextObj::createBuffer** ( unsigned int *type*, RTformat *format*, RTsize *width*, RTsize *height*, RTsize *depth* ) `[inline]`

Create a buffer with given RTbuffertype, RTformat and dimension. See rtBufferCreate, rtBufferSetFormat and rtBufferSetSize3D.

Definition at line 1645 of file optixpp_namespace.h.

#### 1.2.3.14 Buffer **optix::ContextObj::createBufferForCUDA** ( unsigned int *type* ) `[inline]`

Create a buffer for CUDA with given RTbuffertype. See rtBufferCreate.

Definition at line 1654 of file optixpp_namespace.h.

#### 1.2.3.15 Buffer optix::ContextObj::createBufferForCUDA ( unsigned int *type*, RTformat *format* ) [inline]

Create a buffer for CUDA with given RTbuffertype and RTformat. See rtBufferCreate, rtBufferSetFormat.

Definition at line 1661 of file optixpp_namespace.h.

#### 1.2.3.16 Buffer optix::ContextObj::createBufferForCUDA ( unsigned int *type*, RTformat *format*, RTsize *width* ) [inline]

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See rt-BufferCreate, rtBufferSetFormat and rtBufferSetSize1D.

Definition at line 1669 of file optixpp_namespace.h.

#### 1.2.3.17 Buffer optix::ContextObj::createBufferForCUDA ( unsigned int *type*, RTformat *format*, RTsize *width*, RTsize *height* ) [inline]

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See rt-BufferCreate, rtBufferSetFormat and rtBufferSetSize2D.

Definition at line 1678 of file optixpp_namespace.h.

#### 1.2.3.18 Buffer optix::ContextObj::createBufferForCUDA ( unsigned int *type*, RTformat *format*, RTsize *width*, RTsize *height*, RTsize *depth* ) [inline]

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See rt-BufferCreate, rtBufferSetFormat and rtBufferSetSize3D.

Definition at line 1687 of file optixpp_namespace.h.

#### 1.2.3.19 Buffer optix::ContextObj::createBufferFromGLBO ( unsigned int *type*, unsigned int *vbo* ) [inline]

Create buffer from GL buffer object. See rtBufferCreateFromGLBO.

Definition at line 1696 of file optixpp_namespace.h.

#### 1.2.3.20 Geometry optix::ContextObj::createGeometry ( ) [inline]

See rtGeometryCreate.

Definition at line 1771 of file optixpp_namespace.h.

#### 1.2.3.21 GeometryGroup optix::ContextObj::createGeometryGroup ( ) [inline]

See rtGeometryGroupCreate.

Definition at line 1821 of file optixpp_namespace.h.

#### 1.2.3.22 template<class Iterator > GeometryGroup **optix::ContextObj::createGeometryGroup** ( Iterator *childbegin*, Iterator *childend* ) [inline]

Create a GeometryGroup with a set of child nodes. See rtGeometryGroupCreate, rtGeometryGroupSetChildCount and rtGeometryGroupSetChild

Definition at line 1829 of file optixpp_namespace.h.

#### 1.2.3.23 GeometryInstance **optix::ContextObj::createGeometryInstance** ( ) [inline]

See rtGeometryInstanceCreate.

Definition at line 1778 of file optixpp_namespace.h.

#### 1.2.3.24 template<class Iterator > GeometryInstance **optix::ContextObj::createGeometryInstance** ( **Geometry** *geometry*, Iterator *matlbegin*, Iterator *matlend* )

Create a geometry instance with a Geometry object and a set of associated materials. See rtGeometryInstanceCreate, rtGeometryInstanceSetMaterialCount, and rtGeometryInstanceSetMaterial

Definition at line 1786 of file optixpp_namespace.h.

#### 1.2.3.25 Group **optix::ContextObj::createGroup** ( ) [inline]

See rtGroupCreate.

Definition at line 1800 of file optixpp_namespace.h.

#### 1.2.3.26 template<class Iterator > Group **optix::ContextObj::createGroup** ( Iterator *childbegin*, Iterator *childend* ) [inline]

Create a Group with a set of child nodes. See rtGroupCreate, rtGroupSetChildCount and rtGroupSetChild

Definition at line 1808 of file optixpp_namespace.h.

#### 1.2.3.27 Material **optix::ContextObj::createMaterial** ( ) [inline]

See rtMaterialCreate.

Definition at line 1849 of file optixpp_namespace.h.

#### 1.2.3.28 Program **optix::ContextObj::createProgramFromPTXFile** ( const std::string & *ptx*, const std::string & *program_name* ) [inline]

See rtProgramCreateFromPTXFile.

Definition at line 1856 of file optixpp_namespace.h.

#### 1.2.3.29 Program optix::ContextObj::createProgramFromPTXString ( const std::string & *ptx*, const std::string & *program_name* ) [inline]

See rtProgramCreateFromPTXString.

Definition at line 1863 of file optixpp_namespace.h.

#### 1.2.3.30 Selector optix::ContextObj::createSelector ( ) [inline]

See rtSelectorCreate.

Definition at line 1870 of file optixpp_namespace.h.

#### 1.2.3.31 TextureSampler optix::ContextObj::createTextureSampler ( ) [inline]

See rtTextureSamplerCreate.

Definition at line 1877 of file optixpp_namespace.h.

#### 1.2.3.32 TextureSampler optix::ContextObj::createTextureSamplerFromGLImage ( unsigned int *id*, RTgltarget *target* ) [inline]

Create TextureSampler from GL image. See rtTextureSamplerCreateFromGLImage.

Definition at line 1764 of file optixpp_namespace.h.

#### 1.2.3.33 Transform optix::ContextObj::createTransform ( ) [inline]

See rtTransformCreate.

Definition at line 1842 of file optixpp_namespace.h.

#### 1.2.3.34 Variable optix::ContextObj::declareVariable ( const std::string & *name* ) [inline, virtual]

Declare a variable associated with this object. See rt[ObjectType]DeclareVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2125 of file optixpp_namespace.h.

#### 1.2.3.35 Variable optix::ProgramObj::declareVariable ( const std::string & *name* ) [inline, virtual]

Declare a variable associated with this object. See rt[ObjectType]DeclareVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2183 of file optixpp_namespace.h.

#### 1.2.3.36 Variable optix::SelectorObj::declareVariable ( const std::string & *name* ) [inline]

Definition at line 2297 of file optixpp_namespace.h.

#### 1.2.3.37 Variable **optix::GeometryInstanceObj::declareVariable** ( const std::string & *name* ) [inline, virtual]

Declare a variable associated with this object. See rt[ObjectType]DeclareVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2639 of file optixpp_namespace.h.

#### 1.2.3.38 Variable **optix::GeometryObj::declareVariable** ( const std::string & *name* ) [inline, virtual]

Declare a variable associated with this object. See rt[ObjectType]DeclareVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2732 of file optixpp_namespace.h.

#### 1.2.3.39 Variable **optix::MaterialObj::declareVariable** ( const std::string & *name* ) [inline, virtual]

Declare a variable associated with this object. See rt[ObjectType]DeclareVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2825 of file optixpp_namespace.h.

#### 1.2.3.40 void **optix::ContextObj::destroy** ( ) [inline, virtual]

Destroy Context and all of its associated objects. See rtContextDestroy.

Implements optix::DestroyableObj.

Definition at line 1591 of file optixpp_namespace.h.

#### 1.2.3.41 void **optix::ProgramObj::destroy** ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2164 of file optixpp_namespace.h.

#### 1.2.3.42 void **optix::GroupObj::destroy** ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2221 of file optixpp_namespace.h.

#### 1.2.3.43 void **optix::GeometryGroupObj::destroy** ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2378 of file optixpp_namespace.h.

#### 1.2.3.44 void optix::TransformObj::destroy ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2438 of file optixpp_namespace.h.

#### 1.2.3.45 void optix::SelectorObj::destroy ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2240 of file optixpp_namespace.h.

#### 1.2.3.46 void optix::AccelerationObj::destroy ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2486 of file optixpp_namespace.h.

#### 1.2.3.47 void optix::GeometryInstanceObj::destroy ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2575 of file optixpp_namespace.h.

#### 1.2.3.48 void optix::GeometryObj::destroy ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2677 of file optixpp_namespace.h.

#### 1.2.3.49 void optix::MaterialObj::destroy ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2782 of file optixpp_namespace.h.

#### 1.2.3.50 void optix::TextureSamplerObj::destroy ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2863 of file optixpp_namespace.h.

#### 1.2.3.51 void **optix::BufferObj::destroy ( )** `[inline, virtual]`

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 3032 of file optixpp_namespace.h.

#### 1.2.3.52 RTvariable **optix::VariableObj::get ( )** `[inline]`

Get the OptiX C API object wrapped by this instance.

Definition at line 3550 of file optixpp_namespace.h.

#### 1.2.3.53 RTcontext **optix::ContextObj::get ( )** `[inline]`

Return the OptiX C API RTcontext object.

Definition at line 2159 of file optixpp_namespace.h.

#### 1.2.3.54 RTprogram **optix::ProgramObj::get ( )** `[inline]`

Definition at line 2216 of file optixpp_namespace.h.

#### 1.2.3.55 RTgroup **optix::GroupObj::get ( )** `[inline]`

Get the underlying OptiX C API RTgroup opaque pointer.

Definition at line 2373 of file optixpp_namespace.h.

#### 1.2.3.56 RTgeometrygroup **optix::GeometryGroupObj::get ( )** `[inline]`

Get the underlying OptiX C API RTgeometrygroup opaque pointer.

Definition at line 2433 of file optixpp_namespace.h.

#### 1.2.3.57 RTtransform **optix::TransformObj::get ( )** `[inline]`

Get the underlying OptiX C API RTtransform opaque pointer.

Definition at line 2481 of file optixpp_namespace.h.

#### 1.2.3.58 RTselector **optix::SelectorObj::get ( )** `[inline]`

Get the underlying OptiX C API RTselector opaque pointer.

Definition at line 2330 of file optixpp_namespace.h.

#### 1.2.3.59 RTacceleration **optix::AccelerationObj::get ( )** `[inline]`

Get the underlying OptiX C API RTacceleration opaque pointer.

Definition at line 2570 of file optixpp_namespace.h.

#### 1.2.3.60 RTgeometryinstance **optix::GeometryInstanceObj::get ( )** `[inline]`

Get the underlying OptiX C API RTgeometryinstance opaque pointer.

Definition at line 2672 of file optixpp_namespace.h.

#### 1.2.3.61 RTgeometry optix::GeometryObj::get ( ) [inline]

Get the underlying OptiX C API RTgeometry opaque pointer.

Definition at line 2777 of file optixpp_namespace.h.

#### 1.2.3.62 RTmaterial optix::MaterialObj::get ( ) [inline]

Get the underlying OptiX C API RTmaterial opaque pointer.

Definition at line 2858 of file optixpp_namespace.h.

#### 1.2.3.63 RTtexturesampler optix::TextureSamplerObj::get ( ) [inline]

Get the underlying OptiX C API RTtexturesampler opaque pointer.

Definition at line 2983 of file optixpp_namespace.h.

#### 1.2.3.64 RTbuffer optix::BufferObj::get ( ) [inline]

Get the underlying OptiX C API RTbuffer opaque pointer.

Definition at line 3222 of file optixpp_namespace.h.

#### 1.2.3.65 Acceleration optix::GroupObj::getAcceleration ( ) const [inline]

Query the Acceleration structure for this group. See rtGroupGetAcceleration.

Definition at line 2340 of file optixpp_namespace.h.

#### 1.2.3.66 Acceleration optix::GeometryGroupObj::getAcceleration ( ) const [inline]

Query the Acceleration structure for this group. See rtGeometryGroupGetAcceleration.

Definition at line 2402 of file optixpp_namespace.h.

#### 1.2.3.67 std::string optix::VariableObj::getAnnotation ( ) const [inline]

Retrieve the annotation associated with the variable.

Definition at line 3536 of file optixpp_namespace.h.

#### 1.2.3.68 Program optix::MaterialObj::getAnyHitProgram ( unsigned int *ray_type_index* ) const [inline]

Get any hit program for this material at the given *ray_type* index. See rtMaterialGetAny-HitProgram.

Definition at line 2818 of file optixpp_namespace.h.

#### 1.2.3.69 unsigned int optix::TextureSamplerObj::getArraySize ( ) const [inline]

Query the texture array size for this sampler. See rtTextureSamplerGetArraySize.

Definition at line 2899 of file optixpp_namespace.h.

#### 1.2.3.70 RTsize optix::ContextObj::getAvailableDeviceMemory ( int *ordinal* ) const [inline]

See rtContextGetAttribute.

Definition at line 1949 of file optixpp_namespace.h.

#### 1.2.3.71 Program optix::GeometryObj::getBoundingBoxProgram ( ) const [inline]

Get the bounding box program for this geometry. See rtGeometryGetBoundingBox-Program.

Definition at line 2713 of file optixpp_namespace.h.

#### 1.2.3.72 Buffer optix::VariableObj::getBuffer ( ) const [inline]

Definition at line 3521 of file optixpp_namespace.h.

#### 1.2.3.73 Buffer optix::TextureSamplerObj::getBuffer ( unsigned int *texture_array_idx*, unsigned int *mip_level* ) const [inline]

Get the underlying buffer used for texture storage. rtTextureSamplerGetBuffer.

Definition at line 2976 of file optixpp_namespace.h.

#### 1.2.3.74 std::string optix::AccelerationObj::getBuilder ( ) const [inline]

Query the acceleration structure builder. See rtAccelerationGetBuilder.

Definition at line 2534 of file optixpp_namespace.h.

#### 1.2.3.75 template<typename T > T optix::GroupObj::getChild ( unsigned int *index* ) const [inline]

Query an indexed child within this group. See rtGroupGetChild.

Definition at line 2366 of file optixpp_namespace.h.

#### 1.2.3.76 GeometryInstance optix::GeometryGroupObj::getChild ( unsigned int *index* ) const [inline]

Query an indexed GeometryInstance within this group. See rtGeometryGroupGetChild.

Definition at line 2426 of file optixpp_namespace.h.

#### 1.2.3.77 template<typename T > T optix::TransformObj::getChild ( ) const [inline]

Set the child node of this transform. See rtTransformGetChild.

Definition at line 2464 of file optixpp_namespace.h.

#### 1.2.3.78 template<typename T > T optix::SelectorObj::getChild ( unsigned int *index* ) const [inline]

Query an indexed child within this group. See rtSelectorGetChild.

Definition at line 2290 of file optixpp_namespace.h.

#### 1.2.3.79 unsigned int optix::GroupObj::getChildCount ( ) const [inline]

Query the number of children for this group. See rtGroupGetChildCount.

Definition at line 2352 of file optixpp_namespace.h.

#### 1.2.3.80 unsigned int optix::GeometryGroupObj::getChildCount ( ) const [inline]

Query the number of children for this group. See rtGeometryGroupGetChildCount.

Definition at line 2414 of file optixpp_namespace.h.

#### 1.2.3.81 unsigned int optix::SelectorObj::getChildCount ( ) const [inline]

Query the number of children for this group. See rtSelectorGetChildCount.

Definition at line 2276 of file optixpp_namespace.h.

#### 1.2.3.82 Program optix::MaterialObj::getClosestHitProgram ( unsigned int *ray_type_index* ) const [inline]

Get closest hit program for this material at the given *ray_type* index. See rtMaterialGet-ClosestHitProgram.

Definition at line 2806 of file optixpp_namespace.h.

#### 1.2.3.83 Context optix::VariableObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 3227 of file optixpp_namespace.h.

#### 1.2.3.84 Context optix::ContextObj::getContext ( ) const [inline, virtual]

Retrieve the Context object associated with this APIObject. In this case, simply returns itself.

Implements optix::APIObj.

Definition at line 1546 of file optixpp_namespace.h.

#### 1.2.3.85 Context optix::ProgramObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2176 of file optixpp_namespace.h.

#### 1.2.3.86 Context optix::GroupObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2233 of file optixpp_namespace.h.

#### 1.2.3.87 Context optix::GeometryGroupObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2390 of file optixpp_namespace.h.

#### 1.2.3.88 Context optix::TransformObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2450 of file optixpp_namespace.h.

#### 1.2.3.89 Context optix::SelectorObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2252 of file optixpp_namespace.h.

#### 1.2.3.90 Context optix::AccelerationObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2498 of file optixpp_namespace.h.

#### 1.2.3.91 Context optix::GeometryInstanceObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2587 of file optixpp_namespace.h.

#### 1.2.3.92 Context optix::GeometryObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2689 of file optixpp_namespace.h.

#### 1.2.3.93 Context optix::MaterialObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2794 of file optixpp_namespace.h.

#### 1.2.3.94 Context optix::TextureSamplerObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2875 of file optixpp_namespace.h.

#### 1.2.3.95 Context optix::BufferObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 3044 of file optixpp_namespace.h.

#### 1.2.3.96 int optix::ContextObj::getCPUNumThreads ( ) const [inline]

See rtContextGetAttribute.

Definition at line 1921 of file optixpp_namespace.h.

#### 1.2.3.97 void optix::AccelerationObj::getData ( void ∗ *data* ) const [inline]

Get the marshalled acceleration data. See rtAccelerationGetData.

Definition at line 2560 of file optixpp_namespace.h.

#### 1.2.3.98 RTsize optix::AccelerationObj::getDataSize ( ) const [inline]

Query the size of the marshalled acceleration data. See rtAccelerationGetDataSize.

Definition at line 2553 of file optixpp_namespace.h.

#### 1.2.3.99 void optix::ContextObj::getDeviceAttribute ( int *ordinal*, RTdeviceattribute *attrib*, RTsize *size*, void ∗ *p* ) [inline, static]

Call rtDeviceGetAttribute and return the desired attribute value.

Definition at line 1576 of file optixpp_namespace.h.

#### 1.2.3.100 unsigned int **optix::ContextObj::getDeviceCount** ( ) `[inline, static]`

Call rtDeviceGetDeviceCount and returns number of valid devices.

Definition at line 1557 of file optixpp_namespace.h.

#### 1.2.3.101 std::string **optix::ContextObj::getDeviceName** ( int *ordinal* ) `[inline, static]`

Call rtDeviceGetAttribute and return the name of the device.

Definition at line 1566 of file optixpp_namespace.h.

#### 1.2.3.102 void **optix::BufferObj::getDevicePointer** ( unsigned int *optix_device_number,* CUdeviceptr ∗ *device_pointer* ) `[inline]`

Get the pointer to buffer memory on a specific device. See rtBufferGetDevicePointer.

Definition at line 3075 of file optixpp_namespace.h.

#### 1.2.3.103 unsigned int **optix::BufferObj::getDimensionality** ( ) const `[inline]`

Query dimensionality of buffer. See rtBufferGetDimensionality.

Definition at line 3130 of file optixpp_namespace.h.

#### 1.2.3.104 RTsize **optix::BufferObj::getElementSize** ( ) const `[inline]`

Query the data element size for user format buffers. See rtBufferGetElementSize.

Definition at line 3068 of file optixpp_namespace.h.

#### 1.2.3.105 unsigned int **optix::ContextObj::getEnabledDeviceCount** ( ) const `[inline]`

See rtContextGetDeviceCount. As opposed to getDeviceCount, this returns only the number of enabled devices.

Definition at line 1907 of file optixpp_namespace.h.

#### 1.2.3.106 std::vector< int > **optix::ContextObj::getEnabledDevices** ( ) const `[inline]`

See rtContextGetDevices. This returns the list of currently enabled devices.

Definition at line 1899 of file optixpp_namespace.h.

#### 1.2.3.107 unsigned int **optix::ContextObj::getEntryPointCount** ( ) const `[inline]`

See rtContextgetEntryPointCount.

Definition at line 1990 of file optixpp_namespace.h.

#### 1.2.3.108 std::string **optix::ContextObj::getErrorString** ( RTresult *code* ) const [inline]

See rtContextGetErrroString.

Definition at line 1884 of file optixpp_namespace.h.

#### 1.2.3.109 bool **optix::ContextObj::getExceptionEnabled** ( RTexception *exception* ) const [inline]

See rtContextGetExceptionEnabled.

Definition at line 2029 of file optixpp_namespace.h.

#### 1.2.3.110 Program **optix::ContextObj::getExceptionProgram** ( unsigned int *entry_point_index* ) const [inline]

See rtContextGetExceptionProgram.

Definition at line 2016 of file optixpp_namespace.h.

#### 1.2.3.111 void **optix::TextureSamplerObj::getFilteringModes** ( RTfiltermode & *minification,* RTfiltermode & *magnification,* RTfiltermode & *mipmapping* ) const [inline]

Query filtering modes for this sampler. See rtTextureSamplerGetFilteringModes.

Definition at line 2923 of file optixpp_namespace.h.

#### 1.2.3.112 float **optix::VariableObj::getFloat** ( ) const [inline]

Definition at line 3445 of file optixpp_namespace.h.

#### 1.2.3.113 RTformat **optix::BufferObj::getFormat** ( ) const [inline]

Query the data format for the buffer. See rtBufferGetFormat.

Definition at line 3056 of file optixpp_namespace.h.

#### 1.2.3.114 Geometry **optix::GeometryInstanceObj::getGeometry** ( ) const [inline]

Get the geometry object associated with this instance. See rtGeometryInstanceGet-Geometry.

Definition at line 2599 of file optixpp_namespace.h.

#### 1.2.3.115 unsigned int **optix::BufferObj::getGLBOId** ( ) const [inline]

Queries the OpenGL Buffer Object ID associated with this buffer. See rtBufferGetGLB-OId.

Definition at line 3137 of file optixpp_namespace.h.

#### 1.2.3.116 int optix::ContextObj::getGPUPagingActive ( ) const [inline]

See rtContextGetAttribute.

Definition at line 1935 of file optixpp_namespace.h.

#### 1.2.3.117 int optix::ContextObj::getGPUPagingForcedOff ( ) const [inline]

See rtContextGetAttribute.

Definition at line 1942 of file optixpp_namespace.h.

#### 1.2.3.118 int optix::TextureSamplerObj::getId ( ) const [inline]

Returns the device-side ID of this sampler.

Definition at line 2940 of file optixpp_namespace.h.

#### 1.2.3.119 RTtextureindexmode optix::TextureSamplerObj::getIndexingMode ( ) const [inline]

Query texture indexing mode for this sampler. See rtTextureSamplerGetIndexingMode.

Definition at line 2964 of file optixpp_namespace.h.

#### 1.2.3.120 int optix::VariableObj::getInt ( ) const [inline]

Definition at line 3459 of file optixpp_namespace.h.

#### 1.2.3.121 Program optix::GeometryObj::getIntersectionProgram ( ) const [inline]

Get the intersection program for this geometry. See rtGeometryGetIntersection-Program.

Definition at line 2725 of file optixpp_namespace.h.

#### 1.2.3.122 Material optix::GeometryInstanceObj::getMaterial ( unsigned int *idx* ) const [inline]

Get the material at given index. See rtGeometryInstanceGetMaterial.

Definition at line 2623 of file optixpp_namespace.h.

#### 1.2.3.123 unsigned int optix::GeometryInstanceObj::getMaterialCount ( ) const [inline]

Query the number of materials associated with this instance. See rtGeometryInstance-GetMaterialCount.

Definition at line 2611 of file optixpp_namespace.h.

#### 1.2.3.124 void optix::TransformObj::getMatrix ( bool *transpose,* float ∗ *matrix,* float ∗ *inverse_matrix* ) const [inline]

Get the transform matrix for this node. See rtTransformGetMatrix.

Definition at line 2476 of file optixpp_namespace.h.

#### 1.2.3.125 float optix::TextureSamplerObj::getMaxAnisotropy ( ) const [inline]

Query maximum anisotropy for this sampler. See rtTextureSamplerGetMaxAnisotropy.

Definition at line 2933 of file optixpp_namespace.h.

#### 1.2.3.126 int optix::ContextObj::getMaxTextureCount ( ) const [inline]

See rtContextGetAttribute

Definition at line 1914 of file optixpp_namespace.h.

#### 1.2.3.127 unsigned int optix::TextureSamplerObj::getMipLevelCount ( ) const [inline]

Query the number of mip levels for this sampler. See rtTextureSamplerGetMipLevel-Count.

Definition at line 2887 of file optixpp_namespace.h.

#### 1.2.3.128 Program optix::ContextObj::getMissProgram ( unsigned int *ray_type_index* ) const [inline]

See rtContextGetMissProgram.

Definition at line 2054 of file optixpp_namespace.h.

#### 1.2.3.129 std::string optix::VariableObj::getName ( ) const [inline]

Retrieve the name of the variable.

Definition at line 3529 of file optixpp_namespace.h.

#### 1.2.3.130 unsigned int optix::GeometryObj::getPrimitiveCount ( ) const [inline]

Query the number of primitives in this geometry objects (eg, number of triangles in mesh). See rtGeometryGetPrimitiveCount

Definition at line 2701 of file optixpp_namespace.h.

#### 1.2.3.131 RTsize optix::ContextObj::getPrintBufferSize ( ) const [inline]

See rtContextGetPrintBufferSize.

Definition at line 2106 of file optixpp_namespace.h.

#### 1.2.3.132 bool optix::ContextObj::getPrintEnabled ( ) const [inline]

See rtContextGetPrintEnabled.

Definition at line 2094 of file optixpp_namespace.h.

#### 1.2.3.133 optix::int3 optix::ContextObj::getPrintLaunchIndex ( ) const [inline]

See rtContextGetPrintLaunchIndex.

Definition at line 2118 of file optixpp_namespace.h.

#### 1.2.3.134 optix::Program optix::VariableObj::getProgram ( ) const [inline]

Definition at line 3570 of file optixpp_namespace.h.

#### 1.2.3.135 std::string optix::AccelerationObj::getProperty ( const std::string & *name* ) const [inline]

Query properties specifying Acceleration builder/traverser behavior. See rtAcceleration-GetProperty.

Definition at line 2522 of file optixpp_namespace.h.

#### 1.2.3.136 Program optix::ContextObj::getRayGenerationProgram ( unsigned int *entry_point_index* ) const [inline]

See rtContextGetRayGenerationProgram.

Definition at line 2003 of file optixpp_namespace.h.

#### 1.2.3.137 unsigned int optix::ContextObj::getRayTypeCount ( ) const [inline]

See rtContextGetRayTypeCount.

Definition at line 2042 of file optixpp_namespace.h.

#### 1.2.3.138 RTtexturereadmode optix::TextureSamplerObj::getReadMode ( ) const [inline]

Query texture read mode for this sampler. See rtTextureSamplerGetReadMode.

Definition at line 2952 of file optixpp_namespace.h.

#### 1.2.3.139 int optix::ContextObj::getRunningState ( ) const [inline]

See rtContextGetRunningState.

Definition at line 2082 of file optixpp_namespace.h.

#### 1.2.3.140 RTsize optix::VariableObj::getSize ( ) const [inline]

Get the size of the variable data in bytes (eg, float4 returns 4∗sizeof(float) )

Definition at line 3555 of file optixpp_namespace.h.

#### 1.2.3.141 void **optix::BufferObj::getSize** ( RTsize & *width* ) const [inline]

Query 1D buffer dimension. See rtBufferGetSize1D.

Definition at line 3095 of file optixpp_namespace.h.

#### 1.2.3.142 void **optix::BufferObj::getSize** ( RTsize & *width*, RTsize & *height* ) const [inline]

Query 2D buffer dimension. See rtBufferGetSize2D.

Definition at line 3105 of file optixpp_namespace.h.

#### 1.2.3.143 void **optix::BufferObj::getSize** ( RTsize & *width*, RTsize & *height*, RTsize & *depth* ) const [inline]

Query 3D buffer dimension. See rtBufferGetSize3D.

Definition at line 3115 of file optixpp_namespace.h.

#### 1.2.3.144 void **optix::BufferObj::getSize** ( unsigned int *dimensionality*, RTsize $*$ *dims* ) const [inline]

Query dimensions of buffer. See rtBufferGetSizev.

Definition at line 3125 of file optixpp_namespace.h.

#### 1.2.3.145 RTsize **optix::ContextObj::getStackSize** ( ) const [inline]

See rtContextGetStackSize.

Definition at line 1973 of file optixpp_namespace.h.

#### 1.2.3.146 **optix::TextureSampler optix::VariableObj::getTextureSampler** ( ) const [inline]

Definition at line 3562 of file optixpp_namespace.h.

#### 1.2.3.147 std::string **optix::AccelerationObj::getTraverser** ( ) const [inline]

Query the acceleration structure traverser. See rtAccelerationGetTraverser.

Definition at line 2546 of file optixpp_namespace.h.

#### 1.2.3.148 RTobjecttype **optix::VariableObj::getType** ( ) const [inline]

Query the object type of the variable.

Definition at line 3543 of file optixpp_namespace.h.

#### 1.2.3.149 unsigned int **optix::VariableObj::getUint** ( ) const [inline]

Definition at line 3452 of file optixpp_namespace.h.

#### 1.2.3.150 RTsize optix::ContextObj::getUsedHostMemory ( ) const [inline]

See rtContextGetAttribute.

Definition at line 1928 of file optixpp_namespace.h.

#### 1.2.3.151 void optix::VariableObj::getUserData ( RTsize *size*, void ∗ *ptr* ) const [inline]

Retrieve a user defined type given the sizeof the user object.

Definition at line 3481 of file optixpp_namespace.h.

#### 1.2.3.152 Variable optix::ContextObj::getVariable ( unsigned int *index* ) const [inline, virtual]

Query variable by index. See rt[ObjectType]GetVariable.

Implements optix::ScopedObj.

Definition at line 2151 of file optixpp_namespace.h.

#### 1.2.3.153 Variable optix::ProgramObj::getVariable ( unsigned int *index* ) const [inline, virtual]

Query variable by index. See rt[ObjectType]GetVariable.

Implements optix::ScopedObj.

Definition at line 2209 of file optixpp_namespace.h.

#### 1.2.3.154 Variable optix::SelectorObj::getVariable ( unsigned int *index* ) const [inline]

Definition at line 2323 of file optixpp_namespace.h.

#### 1.2.3.155 Variable optix::GeometryInstanceObj::getVariable ( unsigned int *index* ) const [inline, virtual]

Query variable by index. See rt[ObjectType]GetVariable.

Implements optix::ScopedObj.

Definition at line 2665 of file optixpp_namespace.h.

#### 1.2.3.156 Variable optix::GeometryObj::getVariable ( unsigned int *index* ) const [inline, virtual]

Query variable by index. See rt[ObjectType]GetVariable.

Implements optix::ScopedObj.

Definition at line 2758 of file optixpp_namespace.h.

#### 1.2.3.157 Variable optix::MaterialObj::getVariable ( unsigned int *index* ) const [inline, virtual]

Query variable by index. See rt[ObjectType]GetVariable.

Implements optix::ScopedObj.

Definition at line 2851 of file optixpp_namespace.h.

#### 1.2.3.158 unsigned int optix::ContextObj::getVariableCount ( ) const [inline, virtual]

Query the number of variables associated with this object. Used along with ScopedObj::getVariable to iterate over variables in an object. See rt[ObjectType]GetVariableCount

Implements optix::ScopedObj.

Definition at line 2144 of file optixpp_namespace.h.

#### 1.2.3.159 unsigned int optix::ProgramObj::getVariableCount ( ) const [inline, virtual]

Query the number of variables associated with this object. Used along with ScopedObj::getVariable to iterate over variables in an object. See rt[ObjectType]GetVariableCount

Implements optix::ScopedObj.

Definition at line 2202 of file optixpp_namespace.h.

#### 1.2.3.160 unsigned int optix::SelectorObj::getVariableCount ( ) const [inline]

Definition at line 2316 of file optixpp_namespace.h.

#### 1.2.3.161 unsigned int optix::GeometryInstanceObj::getVariableCount ( ) const [inline, virtual]

Query the number of variables associated with this object. Used along with ScopedObj::getVariable to iterate over variables in an object. See rt[ObjectType]GetVariableCount

Implements optix::ScopedObj.

Definition at line 2658 of file optixpp_namespace.h.

#### 1.2.3.162 unsigned int optix::GeometryObj::getVariableCount ( ) const [inline, virtual]

Query the number of variables associated with this object. Used along with ScopedObj::getVariable to iterate over variables in an object. See rt[ObjectType]GetVariableCount

Implements optix::ScopedObj.

Definition at line 2751 of file optixpp_namespace.h.

#### 1.2.3.163 unsigned int **optix::MaterialObj::getVariableCount** ( ) const `[inline, virtual]`

Query the number of variables associated with this object. Used along with ScopedObj::getVariable to iterate over variables in an object. See rt[ObjectType]GetVariableCount

Implements optix::ScopedObj.

Definition at line 2844 of file optixpp_namespace.h.

#### 1.2.3.164 Program **optix::SelectorObj::getVisitProgram** ( ) const `[inline]`

Get the visitor program for this selector. See rtSelectorGetVisitProgram.

Definition at line 2264 of file optixpp_namespace.h.

#### 1.2.3.165 RTwrapmode **optix::TextureSamplerObj::getWrapMode** ( unsigned int *dim* ) const `[inline]`

Query the texture wrap mode for this sampler. See rtTextureSamplerGetWrapMode.

Definition at line 2911 of file optixpp_namespace.h.

#### 1.2.3.166 bool **optix::AccelerationObj::isDirty** ( ) const `[inline]`

Query if the acceleration needs a rebuild. See rtAccelerationIsDirty.

Definition at line 2510 of file optixpp_namespace.h.

#### 1.2.3.167 bool **optix::GeometryObj::isDirty** ( ) const `[inline]`

Query whether this geometry has been marked dirty. See rtGeometryIsDirty.

Definition at line 2770 of file optixpp_namespace.h.

#### 1.2.3.168 void **optix::ContextObj::launch** ( unsigned int *entry_point_index,* RTsize *image_width* ) `[inline]`

See rtContextLaunch1D

Definition at line 2066 of file optixpp_namespace.h.

#### 1.2.3.169 void **optix::ContextObj::launch** ( unsigned int *entry_point_index,* RTsize *image_width,* RTsize *image_height* ) `[inline]`

See rtContextLaunch2D.

Definition at line 2071 of file optixpp_namespace.h.

#### 1.2.3.170 void **optix::ContextObj::launch** ( unsigned int *entry_point_index,* RTsize *image_width,* RTsize *image_height,* RTsize *image_depth* ) `[inline]`

See rtContextLaunch3D.

Definition at line 2076 of file optixpp_namespace.h.

#### 1.2.3.171 Exception **optix::Exception::makeException** ( RTresult *code,* RTcontext *context* ) `[inline, static]`

Helper for creating exceptions from an RTresult code origination from an OptiX C API function call.

Definition at line 262 of file optixpp_namespace.h.

#### 1.2.3.172 Exception **optix::APIObj::makeException** ( RTresult *code,* RTcontext *context* ) `[inline, static]`

For backwards compatability. Use Exception::makeException instead.

Definition at line 317 of file optixpp_namespace.h.

#### 1.2.3.173 void ∗ **optix::BufferObj::map** ( ) `[inline]`

Maps a buffer object for host access. See rtBufferMap.

Definition at line 3209 of file optixpp_namespace.h.

#### 1.2.3.174 void **optix::AccelerationObj::markDirty** ( ) `[inline]`

Mark the acceleration as needing a rebuild. See rtAccelerationMarkDirty.

Definition at line 2505 of file optixpp_namespace.h.

#### 1.2.3.175 void **optix::GeometryObj::markDirty** ( ) `[inline]`

Mark this geometry as dirty, causing rebuild of parent groups acceleration. See rtGeometryMarkDirty.

Definition at line 2765 of file optixpp_namespace.h.

#### 1.2.3.176 void **optix::BufferObj::markDirty** ( ) `[inline]`

Mark the buffer dirty.

Definition at line 3085 of file optixpp_namespace.h.

#### 1.2.3.177 template<class T > Handle< VariableObj > **optix::Handle< T >::operator[]** ( const std::string & *varname* )

Variable access operator. This operator will query the API object for a variable with the given name, creating a new variable instance if necessary. Only valid for ScopedObjs.

Definition at line 598 of file optixpp_namespace.h.

#### 1.2.3.178 template<class T > Handle< VariableObj > **optix::Handle< T >::operator[]** ( const char ∗ *varname* )

Variable access operator. Identical to operator[](const std::string& varname)

Explicitly define char∗ version to avoid ambiguities between builtin operator[](int, char∗) and Handle::operator[]( std::string ). The problem lies in that a Handle can be cast to a bool then to an int which implies that:

```
Context context;
context["var"];
```

can be interpreted as either

```
1["var"]; // Strange but legal way to index into a string (same as "var"[1]
   )
```

or

```
context[ std::string("var") ];
```

Definition at line 607 of file optixpp_namespace.h.

#### 1.2.3.179 Variable optix::ContextObj::queryVariable ( const std::string & *name* ) const [inline, virtual]

Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2132 of file optixpp_namespace.h.

#### 1.2.3.180 Variable optix::ProgramObj::queryVariable ( const std::string & *name* ) const [inline, virtual]

Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2190 of file optixpp_namespace.h.

#### 1.2.3.181 Variable optix::SelectorObj::queryVariable ( const std::string & *name* ) const [inline]

Definition at line 2304 of file optixpp_namespace.h.

#### 1.2.3.182 Variable optix::GeometryInstanceObj::queryVariable ( const std::string & *name* ) const [inline, virtual]

Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2646 of file optixpp_namespace.h.

#### 1.2.3.183 Variable optix::GeometryObj::queryVariable ( const std::string & *name* ) const [inline, virtual]

Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2739 of file optixpp_namespace.h.

#### 1.2.3.184 Variable **optix::MaterialObj::queryVariable** ( const std::string & *name* ) const `[inline, virtual]`

Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2832 of file optixpp_namespace.h.

#### 1.2.3.185 void **optix::BufferObj::registerGLBuffer** ( ) `[inline]`

Declare the buffer as mutable and inaccessible by OptiX. See rtTextureSamplerGL-Register.

Definition at line 3144 of file optixpp_namespace.h.

#### 1.2.3.186 void **optix::TextureSamplerObj::registerGLTexture** ( ) `[inline]`

Declare the texture's buffer as mutable and inaccessible by OptiX. See rtTexture-SamplerGLRegister.

Definition at line 2988 of file optixpp_namespace.h.

#### 1.2.3.187 void **optix::ContextObj::removeVariable** ( **Variable** *v* ) `[inline, virtual]`

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2139 of file optixpp_namespace.h.

#### 1.2.3.188 void **optix::ProgramObj::removeVariable** ( **Variable** *v* ) `[inline, virtual]`

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2197 of file optixpp_namespace.h.

#### 1.2.3.189 void **optix::SelectorObj::removeVariable** ( **Variable** *v* ) `[inline]`

Definition at line 2311 of file optixpp_namespace.h.

#### 1.2.3.190 void **optix::GeometryInstanceObj::removeVariable** ( **Variable** *v* ) `[inline, virtual]`

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2653 of file optixpp_namespace.h.

#### 1.2.3.191 void optix::GeometryObj::removeVariable ( Variable *v* ) [inline, virtual]

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2746 of file optixpp_namespace.h.

#### 1.2.3.192 void optix::MaterialObj::removeVariable ( Variable *v* ) [inline, virtual]

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2839 of file optixpp_namespace.h.

#### 1.2.3.193 void optix::VariableObj::set ( Buffer *buffer* ) [inline]

Definition at line 3471 of file optixpp_namespace.h.

#### 1.2.3.194 void optix::VariableObj::set1fv ( const float ∗ *f* ) [inline]

Set variable value to a scalar float.

Definition at line 3369 of file optixpp_namespace.h.

#### 1.2.3.195 void optix::VariableObj::set1iv ( const int ∗ *i* ) [inline]

Definition at line 3425 of file optixpp_namespace.h.

#### 1.2.3.196 void optix::VariableObj::set1uiv ( const unsigned int ∗ *u* ) [inline]

Definition at line 3269 of file optixpp_namespace.h.

#### 1.2.3.197 void optix::VariableObj::set2fv ( const float ∗ *f* ) [inline]

Set variable value to a float2.

Definition at line 3374 of file optixpp_namespace.h.

#### 1.2.3.198 void optix::VariableObj::set2iv ( const int ∗ *i* ) [inline]

Definition at line 3430 of file optixpp_namespace.h.

#### 1.2.3.199 void optix::VariableObj::set2uiv ( const unsigned int ∗ *u* ) [inline]

Definition at line 3274 of file optixpp_namespace.h.

#### 1.2.3.200 void optix::VariableObj::set3fv ( const float ∗ *f* ) [inline]

Set variable value to a float3.

Definition at line 3379 of file optixpp_namespace.h.

#### 1.2.3.201 void optix::VariableObj::set3iv ( const int ∗ *i* ) [inline]

Definition at line 3435 of file optixpp_namespace.h.

#### 1.2.3.202 void optix::VariableObj::set3uiv ( const unsigned int ∗ *u* ) [inline]

Definition at line 3279 of file optixpp_namespace.h.

#### 1.2.3.203 void optix::VariableObj::set4fv ( const float ∗ *f* ) [inline]

Set variable value to a float4.

Definition at line 3384 of file optixpp_namespace.h.

#### 1.2.3.204 void optix::VariableObj::set4iv ( const int ∗ *i* ) [inline]

Definition at line 3440 of file optixpp_namespace.h.

#### 1.2.3.205 void optix::VariableObj::set4uiv ( const unsigned int ∗ *u* ) [inline]

Definition at line 3284 of file optixpp_namespace.h.

#### 1.2.3.206 void optix::GroupObj::setAcceleration ( Acceleration *acceleration* ) [inline]

Set the Acceleration structure for this group. See rtGroupSetAcceleration.

Definition at line 2335 of file optixpp_namespace.h.

#### 1.2.3.207 void optix::GeometryGroupObj::setAcceleration ( Acceleration *acceleration* ) [inline]

Set the Acceleration structure for this group. See rtGeometryGroupSetAcceleration.

Definition at line 2397 of file optixpp_namespace.h.

#### 1.2.3.208 void optix::MaterialObj::setAnyHitProgram ( unsigned int *ray_type_index*, Program *program* ) [inline]

Set any hit program for this material at the given *ray_type* index. See rtMaterialSetAnyHitProgram.

Definition at line 2813 of file optixpp_namespace.h.

#### 1.2.3.209 void optix::TextureSamplerObj::setArraySize ( unsigned int *num_textures_in_array* ) [inline]

Set the texture array size for this sampler. See rtTextureSamplerSetArraySize.

Definition at line 2894 of file optixpp_namespace.h.

#### 1.2.3.210 void **optix::GeometryObj::setBoundingBoxProgram** ( **Program** *program* ) `[inline]`

Set the bounding box program for this geometry. See rtGeometrySetBoundingBox-Program.

Definition at line 2708 of file optixpp_namespace.h.

#### 1.2.3.211 void **optix::VariableObj::setBuffer** ( **Buffer** *buffer* ) `[inline]`

Definition at line 3466 of file optixpp_namespace.h.

#### 1.2.3.212 void **optix::TextureSamplerObj::setBuffer** ( unsigned int *texture_array_idx*, unsigned int *mip_level*, **Buffer** *buffer* ) `[inline]`

Set the underlying buffer used for texture storage. rtTextureSamplerSetBuffer.

Definition at line 2971 of file optixpp_namespace.h.

#### 1.2.3.213 void **optix::AccelerationObj::setBuilder** ( const std::string & *builder* ) `[inline]`

Specify the acceleration structure builder. See rtAccelerationSetBuilder.

Definition at line 2529 of file optixpp_namespace.h.

#### 1.2.3.214 template<typename T > void **optix::GroupObj::setChild** ( unsigned int *index*, T *child* ) `[inline]`

Set an indexed child within this group. See rtGroupSetChild.

Definition at line 2360 of file optixpp_namespace.h.

#### 1.2.3.215 void **optix::GeometryGroupObj::setChild** ( unsigned int *index*, **GeometryInstance** *geometryinstance* ) `[inline]`

Set an indexed GeometryInstance child of this group. See rtGeometryGroupSetChild.

Definition at line 2421 of file optixpp_namespace.h.

#### 1.2.3.216 template<typename T > void **optix::TransformObj::setChild** ( T *child* ) `[inline]`

Set the child node of this transform. See rtTransformSetChild.

Definition at line 2458 of file optixpp_namespace.h.

#### 1.2.3.217 template<typename T > void **optix::SelectorObj::setChild** ( unsigned int *index*, T *child* ) `[inline]`

Set an indexed child child of this group. See rtSelectorSetChild.

Definition at line 2284 of file optixpp_namespace.h.

#### 1.2.3.218 void **optix::GroupObj::setChildCount** ( unsigned int *count* ) [inline]

Set the number of children for this group. See rtGroupSetChildCount.

Definition at line 2347 of file optixpp_namespace.h.

#### 1.2.3.219 void **optix::GeometryGroupObj::setChildCount** ( unsigned int *count* ) [inline]

Set the number of children for this group. See rtGeometryGroupSetChildCount.

Definition at line 2409 of file optixpp_namespace.h.

#### 1.2.3.220 void **optix::SelectorObj::setChildCount** ( unsigned int *count* ) [inline]

Set the number of children for this group. See rtSelectorSetChildCount.

Definition at line 2271 of file optixpp_namespace.h.

#### 1.2.3.221 void **optix::MaterialObj::setClosestHitProgram** ( unsigned int *ray_type_index*, Program *program* ) [inline]

Set closest hit program for this material at the given *ray_type* index. See rtMaterialSet-ClosestHitProgram.

Definition at line 2801 of file optixpp_namespace.h.

#### 1.2.3.222 void **optix::ContextObj::setCPUNumThreads** ( int *cpu_num_threads* ) [inline]

See rtContextSetAttribute

Definition at line 1958 of file optixpp_namespace.h.

#### 1.2.3.223 void **optix::AccelerationObj::setData** ( const void $*$ *data*, RTsize *size* ) [inline]

Specify the acceleration structure via marshalled acceleration data. See rtAcceleration-SetData.

Definition at line 2565 of file optixpp_namespace.h.

#### 1.2.3.224 void **optix::BufferObj::setDevicePointer** ( unsigned int *optix_device_number*, CUdeviceptr *device_pointer* ) [inline]

Set the pointer to buffer memory on a specific device. See rtBufferSetDevicePointer.

Definition at line 3080 of file optixpp_namespace.h.

#### 1.2.3.225 template$<$class Iterator $>$ void **optix::ContextObj::setDevices** ( Iterator *begin*, Iterator *end* ) [inline]

See rtContextSetDevices

Definition at line 1892 of file optixpp_namespace.h.

#### 1.2.3.226 void **optix::BufferObj::setElementSize** ( RTsize *size_of_element* ) [inline]

Set the data element size for user format buffers. See rtBufferSetElementSize.

Definition at line 3063 of file optixpp_namespace.h.

#### 1.2.3.227 void **optix::ContextObj::setEntryPointCount** ( unsigned int *num_entry_points* ) [inline]

See rtContextSetEntryPointCount.

Definition at line 1985 of file optixpp_namespace.h.

#### 1.2.3.228 void **optix::ContextObj::setExceptionEnabled** ( RTexception *exception*, bool *enabled* ) [inline]

See rtContextSetExceptionEnabled.

Definition at line 2024 of file optixpp_namespace.h.

#### 1.2.3.229 void **optix::ContextObj::setExceptionProgram** ( unsigned int *entry_point_index*, **Program** *program* ) [inline]

See rtContextSetExceptionProgram.

Definition at line 2011 of file optixpp_namespace.h.

#### 1.2.3.230 void **optix::TextureSamplerObj::setFilteringModes** ( RTfiltermode *minification*, RTfiltermode *magnification*, RTfiltermode *mipmapping* ) [inline]

Set filtering modes for this sampler. See rtTextureSamplerSetFilteringModes.

Definition at line 2918 of file optixpp_namespace.h.

#### 1.2.3.231 void **optix::VariableObj::setFloat** ( float *f1* ) [inline]

Set variable value to a scalar float.

Definition at line 3334 of file optixpp_namespace.h.

#### 1.2.3.232 void **optix::VariableObj::setFloat** ( optix::float2 *f* ) [inline]

Set variable value to a float2.

Definition at line 3339 of file optixpp_namespace.h.

#### 1.2.3.233 void **optix::VariableObj::setFloat** ( float *f1*, float *f2* ) [inline]

Set variable value to a float2.

Definition at line 3344 of file optixpp_namespace.h.

#### 1.2.3.234 void **optix::VariableObj::setFloat** ( optix::float3 *f* ) [inline]

Set variable value to a float3.

Definition at line 3349 of file optixpp_namespace.h.

#### 1.2.3.235 void optix::VariableObj::setFloat ( float *f1,* float *f2,* float *f3* ) `[inline]`

Set variable value to a float3.

Definition at line 3354 of file optixpp_namespace.h.

#### 1.2.3.236 void optix::VariableObj::setFloat ( optix::float4 *f* ) `[inline]`

Set variable value to a float4.

Definition at line 3359 of file optixpp_namespace.h.

#### 1.2.3.237 void optix::VariableObj::setFloat ( float *f1,* float *f2,* float *f3,* float *f4* ) `[inline]`

Set variable value to a float4.

Definition at line 3364 of file optixpp_namespace.h.

#### 1.2.3.238 void optix::BufferObj::setFormat ( RTformat *format* ) `[inline]`

Set the data format for the buffer. See rtBufferSetFormat.

Definition at line 3051 of file optixpp_namespace.h.

#### 1.2.3.239 void optix::GeometryInstanceObj::setGeometry ( Geometry *geometry* ) `[inline]`

Set the geometry object associated with this instance. See rtGeometryInstanceSet-Geometry.

Definition at line 2594 of file optixpp_namespace.h.

#### 1.2.3.240 void optix::ContextObj::setGPUPagingForcedOff ( int *gpu_paging_forced_off* ) `[inline]`

See rtContextSetAttribute.

Definition at line 1963 of file optixpp_namespace.h.

#### 1.2.3.241 void optix::TextureSamplerObj::setIndexingMode ( RTtextureindexmode *indexmode* ) `[inline]`

Set texture indexing mode for this sampler. See rtTextureSamplerSetIndexingMode.

Definition at line 2959 of file optixpp_namespace.h.

#### 1.2.3.242 void optix::VariableObj::setInt ( int *i1* ) `[inline]`

Definition at line 3390 of file optixpp_namespace.h.

#### 1.2.3.243 void optix::VariableObj::setInt ( int *i1,* int *i2* ) `[inline]`

Definition at line 3400 of file optixpp_namespace.h.

#### 1.2.3.244 void **optix::VariableObj::setInt** ( optix::int2 *i* ) [inline]

Definition at line 3395 of file optixpp_namespace.h.

#### 1.2.3.245 void **optix::VariableObj::setInt** ( int *i1,* int *i2,* int *i3* ) [inline]

Definition at line 3410 of file optixpp_namespace.h.

#### 1.2.3.246 void **optix::VariableObj::setInt** ( optix::int3 *i* ) [inline]

Definition at line 3405 of file optixpp_namespace.h.

#### 1.2.3.247 void **optix::VariableObj::setInt** ( int *i1,* int *i2,* int *i3,* int *i4* ) [inline]

Definition at line 3420 of file optixpp_namespace.h.

#### 1.2.3.248 void **optix::VariableObj::setInt** ( optix::int4 *i* ) [inline]

Definition at line 3415 of file optixpp_namespace.h.

#### 1.2.3.249 void **optix::GeometryObj::setIntersectionProgram** ( **Program** *program* ) [inline]

Set the intersection program for this geometry. See rtGeometrySetIntersectionProgram.

Definition at line 2720 of file optixpp_namespace.h.

#### 1.2.3.250 void **optix::GeometryInstanceObj::setMaterial** ( unsigned int *idx,* **Material** *material* ) [inline]

Set the material at given index. See rtGeometryInstanceSetMaterial.

Definition at line 2618 of file optixpp_namespace.h.

#### 1.2.3.251 void **optix::GeometryInstanceObj::setMaterialCount** ( unsigned int *count* ) [inline]

Set the number of materials associated with this instance. See rtGeometryInstanceSet-MaterialCount.

Definition at line 2606 of file optixpp_namespace.h.

#### 1.2.3.252 void **optix::TransformObj::setMatrix** ( bool *transpose,* const float ∗ *matrix,* const float ∗ *inverse_matrix* ) [inline]

Set the transform matrix for this node. See rtTransformSetMatrix.

Definition at line 2471 of file optixpp_namespace.h.

#### 1.2.3.253 void **optix::VariableObj::setMatrix2x2fv** ( bool *transpose,* const float ∗ *m* ) [inline]

Definition at line 3289 of file optixpp_namespace.h.

#### 1.2.3.254 void **optix::VariableObj::setMatrix2x3fv** ( bool *transpose*, const float ∗ *m* ) [inline]

Definition at line 3294 of file optixpp_namespace.h.

#### 1.2.3.255 void **optix::VariableObj::setMatrix2x4fv** ( bool *transpose*, const float ∗ *m* ) [inline]

Definition at line 3299 of file optixpp_namespace.h.

#### 1.2.3.256 void **optix::VariableObj::setMatrix3x2fv** ( bool *transpose*, const float ∗ *m* ) [inline]

Definition at line 3304 of file optixpp_namespace.h.

#### 1.2.3.257 void **optix::VariableObj::setMatrix3x3fv** ( bool *transpose*, const float ∗ *m* ) [inline]

Definition at line 3309 of file optixpp_namespace.h.

#### 1.2.3.258 void **optix::VariableObj::setMatrix3x4fv** ( bool *transpose*, const float ∗ *m* ) [inline]

Definition at line 3314 of file optixpp_namespace.h.

#### 1.2.3.259 void **optix::VariableObj::setMatrix4x2fv** ( bool *transpose*, const float ∗ *m* ) [inline]

Definition at line 3319 of file optixpp_namespace.h.

#### 1.2.3.260 void **optix::VariableObj::setMatrix4x3fv** ( bool *transpose*, const float ∗ *m* ) [inline]

Definition at line 3324 of file optixpp_namespace.h.

#### 1.2.3.261 void **optix::VariableObj::setMatrix4x4fv** ( bool *transpose*, const float ∗ *m* ) [inline]

Definition at line 3329 of file optixpp_namespace.h.

#### 1.2.3.262 void **optix::TextureSamplerObj::setMaxAnisotropy** ( float *value* ) [inline]

Set maximum anisotropy for this sampler. See rtTextureSamplerSetMaxAnisotropy.

Definition at line 2928 of file optixpp_namespace.h.

#### 1.2.3.263 void **optix::TextureSamplerObj::setMipLevelCount** ( unsigned int *num_mip_levels* ) [inline]

Set the number of mip levels for this sampler. See rtTextureSamplerSetMipLevelCount.

Definition at line 2882 of file optixpp_namespace.h.

#### 1.2.3.264 void optix::ContextObj::setMissProgram ( unsigned int *ray_type_index*, Program *program* ) [inline]

See rtContextSetMissProgram.

Definition at line 2049 of file optixpp_namespace.h.

#### 1.2.3.265 void optix::GeometryObj::setPrimitiveCount ( unsigned int *num_primitives* ) [inline]

Set the number of primitives in this geometry objects (eg, number of triangles in mesh). See rtGeometrySetPrimitiveCount

Definition at line 2696 of file optixpp_namespace.h.

#### 1.2.3.266 void optix::ContextObj::setPrintBufferSize ( RTsize *buffer_size_bytes* ) [inline]

See rtContextSetPrintBufferSize.

Definition at line 2101 of file optixpp_namespace.h.

#### 1.2.3.267 void optix::ContextObj::setPrintEnabled ( bool *enabled* ) [inline]

See rtContextSetPrintEnabled

Definition at line 2089 of file optixpp_namespace.h.

#### 1.2.3.268 void optix::ContextObj::setPrintLaunchIndex ( int *x*, int *y* = -1, int *z* = -1 ) [inline]

See rtContextSetPrintLaunchIndex.

Definition at line 2113 of file optixpp_namespace.h.

#### 1.2.3.269 void optix::AccelerationObj::setProperty ( const std::string & *name*, const std::string & *value* ) [inline]

Set properties specifying Acceleration builder/traverser behavior. See rtAcceleration-SetProperty.

Definition at line 2517 of file optixpp_namespace.h.

#### 1.2.3.270 void optix::ContextObj::setRayGenerationProgram ( unsigned int *entry_point_index*, Program *program* ) [inline]

See rtContextSetRayGenerationProgram

Definition at line 1998 of file optixpp_namespace.h.

#### 1.2.3.271 void optix::ContextObj::setRayTypeCount ( unsigned int *num_ray_types* ) [inline]

See rtContextSetRayTypeCount.

Definition at line 2037 of file optixpp_namespace.h.

#### 1.2.3.272 void optix::TextureSamplerObj::setReadMode ( RTtexturereadmode *readmode* ) [inline]

Set texture read mode for this sampler. See rtTextureSamplerSetReadMode.

Definition at line 2947 of file optixpp_namespace.h.

#### 1.2.3.273 void optix::BufferObj::setSize ( RTsize *width* ) [inline]

Set buffer dimensionality to one and buffer width to specified width. See rtBufferSet-Size1D.

Definition at line 3090 of file optixpp_namespace.h.

#### 1.2.3.274 void optix::BufferObj::setSize ( RTsize *width,* RTsize *height* ) [inline]

Set buffer dimensionality to two and buffer dimensions to specified width,height. See rtBufferSetSize2D.

Definition at line 3100 of file optixpp_namespace.h.

#### 1.2.3.275 void optix::BufferObj::setSize ( RTsize *width,* RTsize *height,* RTsize *depth* ) [inline]

Set buffer dimensionality to three and buffer dimensions to specified width,height,depth. See rtBufferSetSize3D.

Definition at line 3110 of file optixpp_namespace.h.

#### 1.2.3.276 void optix::BufferObj::setSize ( unsigned int *dimensionality,* const RTsize ∗ *dims* ) [inline]

Set buffer dimensionality and dimensions to specified values. See rtBufferSetSizev.

Definition at line 3120 of file optixpp_namespace.h.

#### 1.2.3.277 void optix::ContextObj::setStackSize ( RTsize *stack_size_bytes* ) [inline]

See rtContextSetStackSize

Definition at line 1968 of file optixpp_namespace.h.

#### 1.2.3.278 void optix::VariableObj::setTextureSampler ( TextureSampler *texturesample* ) [inline]

Definition at line 3486 of file optixpp_namespace.h.

#### 1.2.3.279 void optix::ContextObj::setTimeoutCallback ( RTtimeoutcallback *callback,* double *min_polling_seconds* ) [inline]

See rtContextSetTimeoutCallback RTtimeoutcallback is defined as typedef int (∗R-Ttimeoutcallback)(void).

Definition at line 1980 of file optixpp_namespace.h.

#### 1.2.3.280 void **optix::AccelerationObj::setTraverser** ( const std::string & *traverser* ) `[inline]`

Specify the acceleration structure traverser. See rtAccelerationSetTraverser.

Definition at line 2541 of file optixpp_namespace.h.

#### 1.2.3.281 void **optix::VariableObj::setUint** ( unsigned int *u1* ) `[inline]`

Definition at line 3234 of file optixpp_namespace.h.

#### 1.2.3.282 void **optix::VariableObj::setUint** ( unsigned int *u1*, unsigned int *u2* ) `[inline]`

Definition at line 3239 of file optixpp_namespace.h.

#### 1.2.3.283 void **optix::VariableObj::setUint** ( unsigned int *u1*, unsigned int *u2*, unsigned int *u3* ) `[inline]`

Definition at line 3244 of file optixpp_namespace.h.

#### 1.2.3.284 void **optix::VariableObj::setUint** ( unsigned int *u1*, unsigned int *u2*, unsigned int *u3*, unsigned int *u4* ) `[inline]`

Definition at line 3249 of file optixpp_namespace.h.

#### 1.2.3.285 void **optix::VariableObj::setUint** ( optix::uint2 *u* ) `[inline]`

Definition at line 3254 of file optixpp_namespace.h.

#### 1.2.3.286 void **optix::VariableObj::setUint** ( optix::uint3 *u* ) `[inline]`

Definition at line 3259 of file optixpp_namespace.h.

#### 1.2.3.287 void **optix::VariableObj::setUint** ( optix::uint4 *u* ) `[inline]`

Definition at line 3264 of file optixpp_namespace.h.

#### 1.2.3.288 void **optix::VariableObj::setUserData** ( RTsize *size*, const void ∗ *ptr* ) `[inline]`

Set the variable to a user defined type given the sizeof the user object.

Definition at line 3476 of file optixpp_namespace.h.

#### 1.2.3.289 void **optix::SelectorObj::setVisitProgram** ( **Program** *program* ) `[inline]`

Set the visitor program for this selector. See rtSelectorSetVisitProgram

Definition at line 2259 of file optixpp_namespace.h.

#### 1.2.3.290 void optix::TextureSamplerObj::setWrapMode ( unsigned int *dim*, RTwrapmode *wrapmode* ) `[inline]`

Set the texture wrap mode for this sampler. See rtTextureSamplerSetWrapMode.

Definition at line 2906 of file optixpp_namespace.h.

#### 1.2.3.291 void optix::BufferObj::unmap ( ) `[inline]`

Unmaps a buffer object. See rtBufferUnmap.

Definition at line 3216 of file optixpp_namespace.h.

#### 1.2.3.292 void optix::BufferObj::unregisterGLBuffer ( ) `[inline]`

Unregister the buffer, re-enabling OptiX operations. See rtTextureSamplerGLUnregister.

Definition at line 3149 of file optixpp_namespace.h.

#### 1.2.3.293 void optix::TextureSamplerObj::unregisterGLTexture ( ) `[inline]`

Unregister the texture's buffer, re-enabling OptiX operations. See rtTextureSamplerGL-Unregister.

Definition at line 2993 of file optixpp_namespace.h.

#### 1.2.3.294 void optix::ContextObj::validate ( ) `[inline, virtual]`

See rtContextValidate.

Implements optix::DestroyableObj.

Definition at line 1597 of file optixpp_namespace.h.

#### 1.2.3.295 void optix::ProgramObj::validate ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2171 of file optixpp_namespace.h.

#### 1.2.3.296 void optix::GroupObj::validate ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2228 of file optixpp_namespace.h.

#### 1.2.3.297 void optix::GeometryGroupObj::validate ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2385 of file optixpp_namespace.h.

#### 1.2.3.298 void optix::TransformObj::validate ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2445 of file optixpp_namespace.h.

#### 1.2.3.299 void optix::SelectorObj::validate ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2247 of file optixpp_namespace.h.

#### 1.2.3.300 void optix::AccelerationObj::validate ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2493 of file optixpp_namespace.h.

#### 1.2.3.301 void optix::GeometryInstanceObj::validate ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2582 of file optixpp_namespace.h.

#### 1.2.3.302 void optix::GeometryObj::validate ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2684 of file optixpp_namespace.h.

#### 1.2.3.303 void optix::MaterialObj::validate ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2789 of file optixpp_namespace.h.

#### 1.2.3.304 void optix::TextureSamplerObj::validate ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2870 of file optixpp_namespace.h.

#### 1.2.3.305 void optix::BufferObj::validate ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

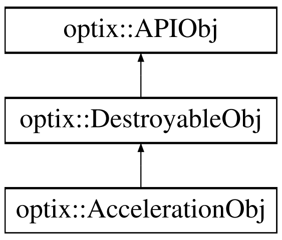
Definition at line 3039 of file optixpp_namespace.h.

# 2 Class Documentation

## 2.1 optix::AccelerationObj Class Reference

```
#include <optixpp_namespace.h>
```

Inheritance diagram for optix::AccelerationObj:



### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- RTacceleration get ()

### Friends

- class Handle< AccelerationObj >

- void markDirty ()
- bool isDirty () const

- void setProperty (const std::string &name, const std::string &value)
- std::string getProperty (const std::string &name) const
- void setBuilder (const std::string &builder)
- std::string getBuilder () const
- void setTraverser (const std::string &traverser)
- std::string getTraverser () const

- RTsize getDataSize () const
- void getData (void ∗data) const
- void setData (const void ∗data, RTsize size)

### 2.1.1 Detailed Description

Acceleration wraps the OptiX C API RTacceleration opaque type and its associated function set.

Definition at line 1101 of file optixpp_namespace.h.

### 2.1.2 Member Function Documentation

#### 2.1.2.1 void **optix::AccelerationObj::destroy** ( ) `[inline, virtual]`

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2486 of file optixpp_namespace.h.

#### 2.1.2.2 RTacceleration **optix::AccelerationObj::get** ( ) `[inline]`

Get the underlying OptiX C API RTacceleration opaque pointer.

Definition at line 2570 of file optixpp_namespace.h.

#### 2.1.2.3 std::string **optix::AccelerationObj::getBuilder** ( ) const `[inline]`

Query the acceleration structure builder. See rtAccelerationGetBuilder.

Definition at line 2534 of file optixpp_namespace.h.

#### 2.1.2.4 Context **optix::AccelerationObj::getContext** ( ) const `[inline, virtual]`

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2498 of file optixpp_namespace.h.

#### 2.1.2.5 void **optix::AccelerationObj::getData** ( void ∗ *data* ) const `[inline]`

Get the marshalled acceleration data. See rtAccelerationGetData.

Definition at line 2560 of file optixpp_namespace.h.

#### 2.1.2.6 RTsize **optix::AccelerationObj::getDataSize** ( ) const `[inline]`

Query the size of the marshalled acceleration data. See rtAccelerationGetDataSize.

Definition at line 2553 of file optixpp_namespace.h.

#### 2.1.2.7 std::string **optix::AccelerationObj::getProperty** ( const std::string & *name* ) const `[inline]`

Query properties specifying Acceleration builder/traverser behavior. See rtAcceleration-GetProperty.

Definition at line 2522 of file optixpp_namespace.h.

#### 2.1.2.8 std::string **optix::AccelerationObj::getTraverser** ( ) const `[inline]`

Query the acceleration structure traverser. See rtAccelerationGetTraverser.

Definition at line 2546 of file optixpp_namespace.h.

#### 2.1.2.9 bool optix::AccelerationObj::isDirty ( ) const [inline]

Query if the acceleration needs a rebuild. See rtAccelerationIsDirty.

Definition at line 2510 of file optixpp_namespace.h.

#### 2.1.2.10 void optix::AccelerationObj::markDirty ( ) [inline]

Mark the acceleration as needing a rebuild. See rtAccelerationMarkDirty.

Definition at line 2505 of file optixpp_namespace.h.

#### 2.1.2.11 void optix::AccelerationObj::setBuilder ( const std::string & *builder* ) [inline]

Specify the acceleration structure builder. See rtAccelerationSetBuilder.

Definition at line 2529 of file optixpp_namespace.h.

#### 2.1.2.12 void optix::AccelerationObj::setData ( const void ∗ *data*, RTsize *size* ) [inline]

Specify the acceleration structure via marshalled acceleration data. See rtAcceleration-SetData.

Definition at line 2565 of file optixpp_namespace.h.

#### 2.1.2.13 void optix::AccelerationObj::setProperty ( const std::string & *name*, const std::string & *value* ) [inline]

Set properties specifying Acceleration builder/traverser behavior. See rtAcceleration-SetProperty.

Definition at line 2517 of file optixpp_namespace.h.

#### 2.1.2.14 void optix::AccelerationObj::setTraverser ( const std::string & *traverser* ) [inline]

Specify the acceleration structure traverser. See rtAccelerationSetTraverser.

Definition at line 2541 of file optixpp_namespace.h.

#### 2.1.2.15 void optix::AccelerationObj::validate ( ) [inline, virtual]

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2493 of file optixpp_namespace.h.

### 2.1.3 Friends And Related Function Documentation

#### 2.1.3.1 friend class Handle< AccelerationObj > [friend]

Definition at line 1149 of file optixpp_namespace.h.

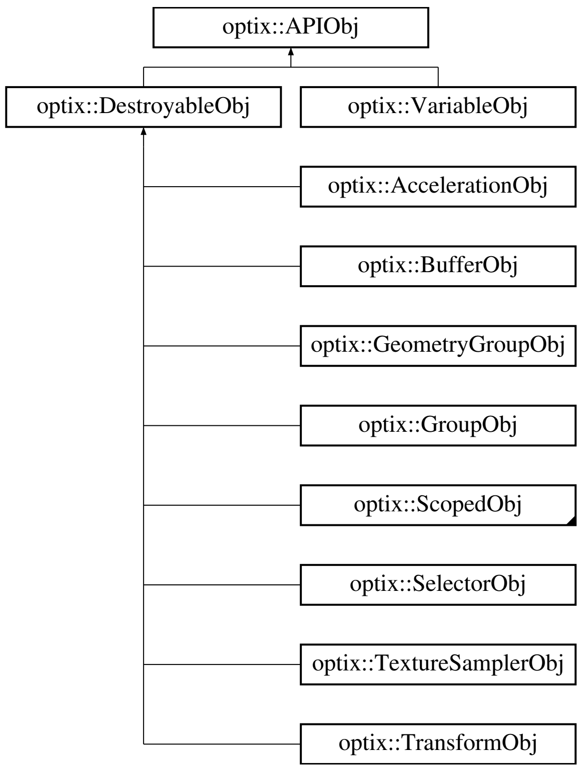
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.2 optix::APIObj Class Reference

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::APIObj:



### Public Member Functions

- APIObj ()
- virtual ∼APIObj ()
- void addReference ()
- int removeReference ()
- virtual Context getContext () const =0
- virtual void checkError (RTresult code) const
- virtual void checkError (RTresult code, Context context) const
- void checkErrorNoGetContext (RTresult code) const

#### Static Public Member Functions

- static Exception makeException (RTresult code, RTcontext context)

### 2.2.1 Detailed Description

Base class for all reference counted wrappers around OptiX C API opaque types.

Wraps:

- RTcontext
- RTbuffer
- RTgeometry
- RTgeometryinstance
- RTgeometrygroup
- RTgroup
- RTmaterial
- RTprogram
- RTselector
- RTtexturesampler
- RTtransform
- RTvariable

Definition at line 291 of file optixpp_namespace.h.

### 2.2.2 Constructor & Destructor Documentation

#### 2.2.2.1 optix::APIObj::APIObj ( ) [inline]

Definition at line 293 of file optixpp_namespace.h.

#### 2.2.2.2 virtual optix::APIObj::∼APIObj ( ) [inline, virtual]

Definition at line 294 of file optixpp_namespace.h.

### 2.2.3 Member Function Documentation

#### 2.2.3.1 void optix::APIObj::addReference ( ) [inline]

Increment the reference count for this object.

Definition at line 297 of file optixpp_namespace.h.

#### 2.2.3.2 void optix::APIObj::checkError ( RTresult *code* ) const [inline, virtual]

Check the given result code and throw an error with appropriate message if the code is not RTsuccess

Reimplemented in optix::ContextObj.

Definition at line 1523 of file optixpp_namespace.h.

#### 2.2.3.3 void optix::APIObj::checkError ( RTresult *code*, Context *context* ) const [inline, virtual]

Definition at line 1531 of file optixpp_namespace.h.

#### 2.2.3.4 void optix::APIObj::checkErrorNoGetContext ( RTresult *code* ) const [inline]

Definition at line 1539 of file optixpp_namespace.h.

#### 2.2.3.5 virtual Context optix::APIObj::getContext ( ) const [pure virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implemented in optix::BufferObj, optix::TextureSamplerObj, optix::MaterialObj, optix::-GeometryObj, optix::GeometryInstanceObj, optix::AccelerationObj, optix::SelectorObj, optix::TransformObj, optix::GeometryGroupObj, optix::GroupObj, optix::ProgramObj, optix::ContextObj, and optix::VariableObj.

#### 2.2.3.6 Exception optix::APIObj::makeException ( RTresult *code*, RTcontext *context* ) [inline, static]

For backwards compatability. Use Exception::makeException instead.

Definition at line 317 of file optixpp_namespace.h.

#### 2.2.3.7 int optix::APIObj::removeReference ( ) [inline]

Decrement the reference count for this object.

Definition at line 299 of file optixpp_namespace.h.

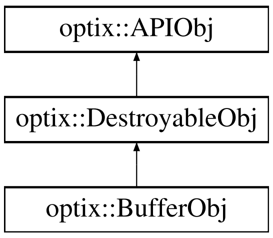
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.3 optix::BufferObj Class Reference

```
#include <optixpp_namespace.h>
```

Inheritance diagram for optix::BufferObj:



### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- RTbuffer get ()

### Friends

- class Handle< BufferObj >

- void setFormat (RTformat format)
- RTformat getFormat () const
- void setElementSize (RTsize size_of_element)
- RTsize getElementSize () const
- void getDevicePointer (unsigned int optix_device_number, CUdeviceptr ∗device-_pointer)
- void setDevicePointer (unsigned int optix_device_number, CUdeviceptr device_-pointer)
- void markDirty ()
- void setSize (RTsize width)
- void getSize (RTsize &width) const
- void setSize (RTsize width, RTsize height)
- void getSize (RTsize &width, RTsize &height) const
- void setSize (RTsize width, RTsize height, RTsize depth)
- void getSize (RTsize &width, RTsize &height, RTsize &depth) const
- void setSize (unsigned int dimensionality, const RTsize ∗dims)
- void getSize (unsigned int dimensionality, RTsize ∗dims) const
- unsigned int getDimensionality () const

- unsigned int getGLBOId () const
- void registerGLBuffer ()
- void unregisterGLBuffer ()

- void ∗ map ()
- void unmap ()

### 2.3.1 Detailed Description

Buffer wraps the OptiX C API RTbuffer opaque type and its associated function set.

Definition at line 1415 of file optixpp_namespace.h.

### 2.3.2 Member Function Documentation

#### 2.3.2.1 void **optix::BufferObj::destroy** ( ) `[inline, virtual]`

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 3032 of file optixpp_namespace.h.

#### 2.3.2.2 RTbuffer **optix::BufferObj::get** ( ) `[inline]`

Get the underlying OptiX C API RTbuffer opaque pointer.

Definition at line 3222 of file optixpp_namespace.h.

#### 2.3.2.3 Context **optix::BufferObj::getContext** ( ) const `[inline, virtual]`

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 3044 of file optixpp_namespace.h.

#### 2.3.2.4 void **optix::BufferObj::getDevicePointer** ( unsigned int *optix_device_number,* CUdeviceptr ∗ *device_pointer* ) `[inline]`

Get the pointer to buffer memory on a specific device. See rtBufferGetDevicePointer.

Definition at line 3075 of file optixpp_namespace.h.

#### 2.3.2.5 unsigned int **optix::BufferObj::getDimensionality** ( ) const `[inline]`

Query dimensionality of buffer. See rtBufferGetDimensionality.

Definition at line 3130 of file optixpp_namespace.h.

#### 2.3.2.6 RTsize **optix::BufferObj::getElementSize** ( ) const `[inline]`

Query the data element size for user format buffers. See rtBufferGetElementSize.

Definition at line 3068 of file optixpp_namespace.h.

#### 2.3.2.7 RTformat **optix::BufferObj::getFormat** ( ) const `[inline]`

Query the data format for the buffer. See rtBufferGetFormat.

Definition at line 3056 of file optixpp_namespace.h.

#### 2.3.2.8 unsigned int **optix::BufferObj::getGLBOId** ( ) const [inline]

Queries the OpenGL Buffer Object ID associated with this buffer. See rtBufferGetGLBOId.

Definition at line 3137 of file optixpp_namespace.h.

#### 2.3.2.9 void **optix::BufferObj::getSize** ( RTsize & *width* ) const [inline]

Query 1D buffer dimension. See rtBufferGetSize1D.

Definition at line 3095 of file optixpp_namespace.h.

#### 2.3.2.10 void **optix::BufferObj::getSize** ( RTsize & *width,* RTsize & *height* ) const [inline]

Query 2D buffer dimension. See rtBufferGetSize2D.

Definition at line 3105 of file optixpp_namespace.h.

#### 2.3.2.11 void **optix::BufferObj::getSize** ( RTsize & *width,* RTsize & *height,* RTsize & *depth* ) const [inline]

Query 3D buffer dimension. See rtBufferGetSize3D.

Definition at line 3115 of file optixpp_namespace.h.

#### 2.3.2.12 void **optix::BufferObj::getSize** ( unsigned int *dimensionality,* RTsize ∗ *dims* ) const [inline]

Query dimensions of buffer. See rtBufferGetSizev.

Definition at line 3125 of file optixpp_namespace.h.

#### 2.3.2.13 void ∗ **optix::BufferObj::map** ( ) [inline]

Maps a buffer object for host access. See rtBufferMap.

Definition at line 3209 of file optixpp_namespace.h.

#### 2.3.2.14 void **optix::BufferObj::markDirty** ( ) [inline]

Mark the buffer dirty.

Definition at line 3085 of file optixpp_namespace.h.

#### 2.3.2.15 void **optix::BufferObj::registerGLBuffer** ( ) [inline]

Declare the buffer as mutable and inaccessible by OptiX. See rtTextureSamplerGLRegister.

Definition at line 3144 of file optixpp_namespace.h.

#### 2.3.2.16 void optix::BufferObj::setDevicePointer ( unsigned int *optix_device_number,* CUdeviceptr *device_pointer* ) `[inline]`

Set the pointer to buffer memory on a specific device. See rtBufferSetDevicePointer.

Definition at line 3080 of file optixpp_namespace.h.

#### 2.3.2.17 void optix::BufferObj::setElementSize ( RTsize *size_of_element* ) `[inline]`

Set the data element size for user format buffers. See rtBufferSetElementSize.

Definition at line 3063 of file optixpp_namespace.h.

#### 2.3.2.18 void optix::BufferObj::setFormat ( RTformat *format* ) `[inline]`

Set the data format for the buffer. See rtBufferSetFormat.

Definition at line 3051 of file optixpp_namespace.h.

#### 2.3.2.19 void optix::BufferObj::setSize ( RTsize *width* ) `[inline]`

Set buffer dimensionality to one and buffer width to specified width. See rtBufferSet-Size1D.

Definition at line 3090 of file optixpp_namespace.h.

#### 2.3.2.20 void optix::BufferObj::setSize ( RTsize *width,* RTsize *height* ) `[inline]`

Set buffer dimensionality to two and buffer dimensions to specified width,height. See rtBufferSetSize2D.

Definition at line 3100 of file optixpp_namespace.h.

#### 2.3.2.21 void optix::BufferObj::setSize ( RTsize *width,* RTsize *height,* RTsize *depth* ) `[inline]`

Set buffer dimensionality to three and buffer dimensions to specified width,height,depth. See rtBufferSetSize3D.

Definition at line 3110 of file optixpp_namespace.h.

#### 2.3.2.22 void optix::BufferObj::setSize ( unsigned int *dimensionality,* const RTsize ∗ *dims* ) `[inline]`

Set buffer dimensionality and dimensions to specified values. See rtBufferSetSizev.

Definition at line 3120 of file optixpp_namespace.h.

#### 2.3.2.23 void optix::BufferObj::unmap ( ) `[inline]`

Unmaps a buffer object. See rtBufferUnmap.

Definition at line 3216 of file optixpp_namespace.h.

#### 2.3.2.24 void optix::BufferObj::unregisterGLBuffer ( ) [inline]

Unregister the buffer, re-enabling OptiX operations. See rtTextureSamplerGLUnregister.

Definition at line 3149 of file optixpp_namespace.h.

#### 2.3.2.25 void optix::BufferObj::validate ( ) [inline, virtual]

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 3039 of file optixpp_namespace.h.

### 2.3.3 Friends And Related Function Documentation

#### 2.3.3.1 friend class Handle< BufferObj > [friend]

Definition at line 1516 of file optixpp_namespace.h.

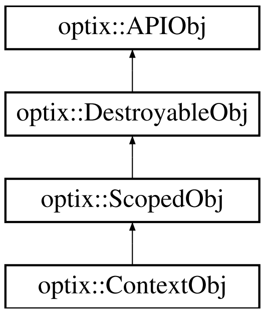
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.4 optix::ContextObj Class Reference

```
#include <optixpp_namespace.h>
```

Inheritance diagram for optix::ContextObj:



### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- void compile ()
- int getRunningState () const
- RTcontext get ()

### Static Public Member Functions

- static unsigned int getDeviceCount ()
- static std::string getDeviceName (int ordinal)
- static void getDeviceAttribute (int ordinal, RTdeviceattribute attrib, RTsize size, void *p)
- static Context create ()

### Friends

- class Handle< ContextObj >

- void checkError (RTresult code) const
- std::string getErrorString (RTresult code) const

- Acceleration createAcceleration (const char *builder, const char *traverser)
- Buffer createBuffer (unsigned int type)
- Buffer createBuffer (unsigned int type, RTformat format)
- Buffer createBuffer (unsigned int type, RTformat format, RTsize width)
- Buffer createBuffer (unsigned int type, RTformat format, RTsize width, RTsize height)
- Buffer createBuffer (unsigned int type, RTformat format, RTsize width, RTsize height, RTsize depth)
- Buffer createBufferForCUDA (unsigned int type)
- Buffer createBufferForCUDA (unsigned int type, RTformat format)
- Buffer createBufferForCUDA (unsigned int type, RTformat format, RTsize width)
- Buffer createBufferForCUDA (unsigned int type, RTformat format, RTsize width, RTsize height)
- Buffer createBufferForCUDA (unsigned int type, RTformat format, RTsize width, RTsize height, RTsize depth)
- Buffer createBufferFromGLBO (unsigned int type, unsigned int vbo)
- TextureSampler createTextureSamplerFromGLImage (unsigned int id, RTgltarget target)
- Geometry createGeometry ()
- GeometryInstance createGeometryInstance ()
- template<class Iterator >
  GeometryInstance createGeometryInstance (Geometry geometry, Iterator matlbegin, Iterator matlend)
- Group createGroup ()
- template<class Iterator >
  Group createGroup (Iterator childbegin, Iterator childend)
- GeometryGroup createGeometryGroup ()
- template<class Iterator >
  GeometryGroup createGeometryGroup (Iterator childbegin, Iterator childend)
- Transform createTransform ()
- Material createMaterial ()

- Program createProgramFromPTXFile (const std::string &ptx, const std::string &program_name)
- Program createProgramFromPTXString (const std::string &ptx, const std::string &program_name)
- Selector createSelector ()
- TextureSampler createTextureSampler ()

- template<class Iterator >
  void setDevices (Iterator begin, Iterator end)
- std::vector< int > getEnabledDevices () const
- unsigned int getEnabledDeviceCount () const

- int getMaxTextureCount () const
- int getCPUNumThreads () const
- RTsize getUsedHostMemory () const
- int getGPUPagingActive () const
- int getGPUPagingForcedOff () const
- RTsize getAvailableDeviceMemory (int ordinal) const

- void setCPUNumThreads (int cpu_num_threads)
- void setGPUPagingForcedOff (int gpu_paging_forced_off)

- void setStackSize (RTsize stack_size_bytes)
- RTsize getStackSize () const
- void setTimeoutCallback (RTtimeoutcallback callback, double min_polling_-seconds)
- void setEntryPointCount (unsigned int num_entry_points)
- unsigned int getEntryPointCount () const
- void setRayTypeCount (unsigned int num_ray_types)
- unsigned int getRayTypeCount () const

- void setRayGenerationProgram (unsigned int entry_point_index, Program program)
- Program getRayGenerationProgram (unsigned int entry_point_index) const
- void setExceptionProgram (unsigned int entry_point_index, Program program)
- Program getExceptionProgram (unsigned int entry_point_index) const
- void setExceptionEnabled (RTexception exception, bool enabled)
- bool getExceptionEnabled (RTexception exception) const
- void setMissProgram (unsigned int ray_type_index, Program program)
- Program getMissProgram (unsigned int ray_type_index) const

- void launch (unsigned int entry_point_index, RTsize image_width)
- void launch (unsigned int entry_point_index, RTsize image_width, RTsize image-_height)
- void launch (unsigned int entry_point_index, RTsize image_width, RTsize image-_height, RTsize image_depth)

- void setPrintEnabled (bool enabled)
- bool getPrintEnabled () const
- void setPrintBufferSize (RTsize buffer_size_bytes)
- RTsize getPrintBufferSize () const
- void setPrintLaunchIndex (int x, int y=-1, int z=-1)
- optix::int3 getPrintLaunchIndex () const

- Variable declareVariable (const std::string &name)
- Variable queryVariable (const std::string &name) const
- void removeVariable (Variable v)
- unsigned int getVariableCount () const
- Variable getVariable (unsigned int index) const

### 2.4.1 Detailed Description

Context object wraps the OptiX C API RTcontext opaque type and its associated function set.

Definition at line 619 of file optixpp_namespace.h.

### 2.4.2 Member Function Documentation

#### 2.4.2.1 void optix::ContextObj::checkError ( RTresult *code* ) const [inline, virtual]

See APIObj::checkError

Reimplemented from optix::APIObj.

Definition at line 1551 of file optixpp_namespace.h.

#### 2.4.2.2 void optix::ContextObj::compile ( ) [inline]

See rtContextCompile.

Definition at line 2061 of file optixpp_namespace.h.

#### 2.4.2.3 Context optix::ContextObj::create ( ) [inline, static]

Creates a Context object. See rtContextCreate.

Definition at line 1582 of file optixpp_namespace.h.

#### 2.4.2.4 Acceleration optix::ContextObj::createAcceleration ( const char ∗ *builder*, const char ∗ *traverser* ) [inline]

See rtAccelerationCreate

Definition at line 1602 of file optixpp_namespace.h.

#### 2.4.2.5 Buffer optix::ContextObj::createBuffer ( unsigned int *type* ) [inline]

Create a buffer with given RTbuffertype. See rtBufferCreate.

Definition at line 1612 of file optixpp_namespace.h.

#### 2.4.2.6 Buffer optix::ContextObj::createBuffer ( unsigned int *type,* RTformat *format* ) [inline]

Create a buffer with given RTbuffertype and RTformat. See rtBufferCreate, rtBufferSet-Format.

Definition at line 1619 of file optixpp_namespace.h.

#### 2.4.2.7 Buffer optix::ContextObj::createBuffer ( unsigned int *type,* RTformat *format,* RTsize *width* ) [inline]

Create a buffer with given RTbuffertype, RTformat and dimension. See rtBufferCreate, rtBufferSetFormat and rtBufferSetSize1D.

Definition at line 1627 of file optixpp_namespace.h.

#### 2.4.2.8 Buffer optix::ContextObj::createBuffer ( unsigned int *type,* RTformat *format,* RTsize *width,* RTsize *height* ) [inline]

Create a buffer with given RTbuffertype, RTformat and dimension. See rtBufferCreate, rtBufferSetFormat and rtBufferSetSize2D.

Definition at line 1636 of file optixpp_namespace.h.

#### 2.4.2.9 Buffer optix::ContextObj::createBuffer ( unsigned int *type,* RTformat *format,* RTsize *width,* RTsize *height,* RTsize *depth* ) [inline]

Create a buffer with given RTbuffertype, RTformat and dimension. See rtBufferCreate, rtBufferSetFormat and rtBufferSetSize3D.

Definition at line 1645 of file optixpp_namespace.h.

#### 2.4.2.10 Buffer optix::ContextObj::createBufferForCUDA ( unsigned int *type* ) [inline]

Create a buffer for CUDA with given RTbuffertype. See rtBufferCreate.

Definition at line 1654 of file optixpp_namespace.h.

#### 2.4.2.11 Buffer optix::ContextObj::createBufferForCUDA ( unsigned int *type,* RTformat *format* ) [inline]

Create a buffer for CUDA with given RTbuffertype and RTformat. See rtBufferCreate, rtBufferSetFormat.

Definition at line 1661 of file optixpp_namespace.h.

#### 2.4.2.12 Buffer optix::ContextObj::createBufferForCUDA ( unsigned int *type*, RTformat *format*, RTsize *width* ) [inline]

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See rt-BufferCreate, rtBufferSetFormat and rtBufferSetSize1D.

Definition at line 1669 of file optixpp_namespace.h.

#### 2.4.2.13 Buffer optix::ContextObj::createBufferForCUDA ( unsigned int *type*, RTformat *format*, RTsize *width*, RTsize *height* ) [inline]

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See rt-BufferCreate, rtBufferSetFormat and rtBufferSetSize2D.

Definition at line 1678 of file optixpp_namespace.h.

#### 2.4.2.14 Buffer optix::ContextObj::createBufferForCUDA ( unsigned int *type*, RTformat *format*, RTsize *width*, RTsize *height*, RTsize *depth* ) [inline]

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See rt-BufferCreate, rtBufferSetFormat and rtBufferSetSize3D.

Definition at line 1687 of file optixpp_namespace.h.

#### 2.4.2.15 Buffer optix::ContextObj::createBufferFromGLBO ( unsigned int *type*, unsigned int *vbo* ) [inline]

Create buffer from GL buffer object. See rtBufferCreateFromGLBO.

Definition at line 1696 of file optixpp_namespace.h.

#### 2.4.2.16 Geometry optix::ContextObj::createGeometry ( ) [inline]

See rtGeometryCreate.

Definition at line 1771 of file optixpp_namespace.h.

#### 2.4.2.17 GeometryGroup optix::ContextObj::createGeometryGroup ( ) [inline]

See rtGeometryGroupCreate.

Definition at line 1821 of file optixpp_namespace.h.

#### 2.4.2.18 template<class Iterator > GeometryGroup optix::ContextObj::createGeometryGroup ( Iterator *childbegin*, Iterator *childend* ) [inline]

Create a GeometryGroup with a set of child nodes. See rtGeometryGroupCreate, rt-GeometryGroupSetChildCount and rtGeometryGroupSetChild

Definition at line 1829 of file optixpp_namespace.h.

#### 2.4.2.19 GeometryInstance optix::ContextObj::createGeometryInstance ( ) [inline]

See rtGeometryInstanceCreate.

Definition at line 1778 of file optixpp_namespace.h.

#### 2.4.2.20 template<class Iterator > GeometryInstance optix::ContextObj::createGeometryInstance ( Geometry *geometry,* Iterator *matlbegin,* Iterator *matlend* )

Create a geometry instance with a Geometry object and a set of associated materials. See rtGeometryInstanceCreate, rtGeometryInstanceSetMaterialCount, and rtGeometryInstanceSetMaterial

Definition at line 1786 of file optixpp_namespace.h.

#### 2.4.2.21 Group optix::ContextObj::createGroup ( ) [inline]

See rtGroupCreate.

Definition at line 1800 of file optixpp_namespace.h.

#### 2.4.2.22 template<class Iterator > Group optix::ContextObj::createGroup ( Iterator *childbegin,* Iterator *childend* ) [inline]

Create a Group with a set of child nodes. See rtGroupCreate, rtGroupSetChildCount and rtGroupSetChild

Definition at line 1808 of file optixpp_namespace.h.

#### 2.4.2.23 Material optix::ContextObj::createMaterial ( ) [inline]

See rtMaterialCreate.

Definition at line 1849 of file optixpp_namespace.h.

#### 2.4.2.24 Program optix::ContextObj::createProgramFromPTXFile ( const std::string & *ptx,* const std::string & *program_name* ) [inline]

See rtProgramCreateFromPTXFile.

Definition at line 1856 of file optixpp_namespace.h.

#### 2.4.2.25 Program optix::ContextObj::createProgramFromPTXString ( const std::string & *ptx,* const std::string & *program_name* ) [inline]

See rtProgramCreateFromPTXString.

Definition at line 1863 of file optixpp_namespace.h.

#### 2.4.2.26 Selector optix::ContextObj::createSelector ( ) [inline]

See rtSelectorCreate.

Definition at line 1870 of file optixpp_namespace.h.

#### 2.4.2.27 TextureSampler optix::ContextObj::createTextureSampler ( ) [inline]

See rtTextureSamplerCreate.

Definition at line 1877 of file optixpp_namespace.h.

#### 2.4.2.28 TextureSampler optix::ContextObj::createTextureSamplerFromGLImage ( unsigned int *id*, RTgltarget *target* ) [inline]

Create TextureSampler from GL image. See rtTextureSamplerCreateFromGLImage.

Definition at line 1764 of file optixpp_namespace.h.

#### 2.4.2.29 Transform optix::ContextObj::createTransform ( ) [inline]

See rtTransformCreate.

Definition at line 1842 of file optixpp_namespace.h.

#### 2.4.2.30 Variable optix::ContextObj::declareVariable ( const std::string & *name* ) [inline, virtual]

Declare a variable associated with this object. See rt[ObjectType]DeclareVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2125 of file optixpp_namespace.h.

#### 2.4.2.31 void optix::ContextObj::destroy ( ) [inline, virtual]

Destroy Context and all of its associated objects. See rtContextDestroy.

Implements optix::DestroyableObj.

Definition at line 1591 of file optixpp_namespace.h.

#### 2.4.2.32 RTcontext optix::ContextObj::get ( ) [inline]

Return the OptiX C API RTcontext object.

Definition at line 2159 of file optixpp_namespace.h.

#### 2.4.2.33 RTsize optix::ContextObj::getAvailableDeviceMemory ( int *ordinal* ) const [inline]

See rtContextGetAttribute.

Definition at line 1949 of file optixpp_namespace.h.

#### 2.4.2.34 Context optix::ContextObj::getContext ( ) const [inline, virtual]

Retrieve the Context object associated with this APIObject. In this case, simply returns itself.

Implements optix::APIObj.

Definition at line 1546 of file optixpp_namespace.h.

#### 2.4.2.35 int optix::ContextObj::getCPUNumThreads ( ) const [inline]

See rtContextGetAttribute.

Definition at line 1921 of file optixpp_namespace.h.

#### 2.4.2.36 void optix::ContextObj::getDeviceAttribute ( int *ordinal*, RTdeviceattribute *attrib*, RTsize *size*, void ∗ *p* ) [inline, static]

Call rtDeviceGetAttribute and return the desired attribute value.

Definition at line 1576 of file optixpp_namespace.h.

#### 2.4.2.37 unsigned int optix::ContextObj::getDeviceCount ( ) [inline, static]

Call rtDeviceGetDeviceCount and returns number of valid devices.

Definition at line 1557 of file optixpp_namespace.h.

#### 2.4.2.38 std::string optix::ContextObj::getDeviceName ( int *ordinal* ) [inline, static]

Call rtDeviceGetAttribute and return the name of the device.

Definition at line 1566 of file optixpp_namespace.h.

#### 2.4.2.39 unsigned int optix::ContextObj::getEnabledDeviceCount ( ) const [inline]

See rtContextGetDeviceCount. As opposed to getDeviceCount, this returns only the number of enabled devices.

Definition at line 1907 of file optixpp_namespace.h.

#### 2.4.2.40 std::vector< int > optix::ContextObj::getEnabledDevices ( ) const [inline]

See rtContextGetDevices. This returns the list of currently enabled devices.

Definition at line 1899 of file optixpp_namespace.h.

#### 2.4.2.41 unsigned int optix::ContextObj::getEntryPointCount ( ) const [inline]

See rtContextgetEntryPointCount.

Definition at line 1990 of file optixpp_namespace.h.

#### 2.4.2.42 std::string optix::ContextObj::getErrorString ( RTresult *code* ) const [inline]

See rtContextGetErrroString.

Definition at line 1884 of file optixpp_namespace.h.

#### 2.4.2.43 bool optix::ContextObj::getExceptionEnabled ( RTexception *exception* ) const [inline]

See rtContextGetExceptionEnabled.

Definition at line 2029 of file optixpp_namespace.h.

#### 2.4.2.44 Program optix::ContextObj::getExceptionProgram ( unsigned int *entry_point_index* ) const [inline]

See rtContextGetExceptionProgram.

Definition at line 2016 of file optixpp_namespace.h.

#### 2.4.2.45 int optix::ContextObj::getGPUPagingActive ( ) const [inline]

See rtContextGetAttribute.

Definition at line 1935 of file optixpp_namespace.h.

#### 2.4.2.46 int optix::ContextObj::getGPUPagingForcedOff ( ) const [inline]

See rtContextGetAttribute.

Definition at line 1942 of file optixpp_namespace.h.

#### 2.4.2.47 int optix::ContextObj::getMaxTextureCount ( ) const [inline]

See rtContextGetAttribute

Definition at line 1914 of file optixpp_namespace.h.

#### 2.4.2.48 Program optix::ContextObj::getMissProgram ( unsigned int *ray_type_index* ) const [inline]

See rtContextGetMissProgram.

Definition at line 2054 of file optixpp_namespace.h.

#### 2.4.2.49 RTsize optix::ContextObj::getPrintBufferSize ( ) const [inline]

See rtContextGetPrintBufferSize.

Definition at line 2106 of file optixpp_namespace.h.

#### 2.4.2.50 bool optix::ContextObj::getPrintEnabled ( ) const [inline]

See rtContextGetPrintEnabled.

Definition at line 2094 of file optixpp_namespace.h.

#### 2.4.2.51 optix::int3 optix::ContextObj::getPrintLaunchIndex ( ) const [inline]

See rtContextGetPrintLaunchIndex.

Definition at line 2118 of file optixpp_namespace.h.

#### 2.4.2.52 Program optix::ContextObj::getRayGenerationProgram ( unsigned int *entry_point_index* ) const [inline]

See rtContextGetRayGenerationProgram.

Definition at line 2003 of file optixpp_namespace.h.

#### 2.4.2.53 unsigned int optix::ContextObj::getRayTypeCount ( ) const [inline]

See rtContextGetRayTypeCount.

Definition at line 2042 of file optixpp_namespace.h.

#### 2.4.2.54 int optix::ContextObj::getRunningState ( ) const [inline]

See rtContextGetRunningState.

Definition at line 2082 of file optixpp_namespace.h.

#### 2.4.2.55 RTsize optix::ContextObj::getStackSize ( ) const [inline]

See rtContextGetStackSize.

Definition at line 1973 of file optixpp_namespace.h.

#### 2.4.2.56 RTsize optix::ContextObj::getUsedHostMemory ( ) const [inline]

See rtContextGetAttribute.

Definition at line 1928 of file optixpp_namespace.h.

#### 2.4.2.57 Variable optix::ContextObj::getVariable ( unsigned int *index* ) const [inline, virtual]

Query variable by index. See rt[ObjectType]GetVariable.

Implements optix::ScopedObj.

Definition at line 2151 of file optixpp_namespace.h.

#### 2.4.2.58 unsigned int optix::ContextObj::getVariableCount ( ) const [inline, virtual]

Query the number of variables associated with this object. Used along with ScopedObj-::getVariable to iterate over variables in an object. See rt[ObjectType]GetVariableCount

Implements optix::ScopedObj.

Definition at line 2144 of file optixpp_namespace.h.

#### 2.4.2.59 void optix::ContextObj::launch ( unsigned int *entry_point_index,* RTsize *image_width* ) [inline]

See rtContextLaunch1D

Definition at line 2066 of file optixpp_namespace.h.

#### 2.4.2.60 void optix::ContextObj::launch ( unsigned int *entry_point_index*, RTsize *image_width*, RTsize *image_height* ) `[inline]`

See rtContextLaunch2D.

Definition at line 2071 of file optixpp_namespace.h.

#### 2.4.2.61 void optix::ContextObj::launch ( unsigned int *entry_point_index*, RTsize *image_width*, RTsize *image_height*, RTsize *image_depth* ) `[inline]`

See rtContextLaunch3D.

Definition at line 2076 of file optixpp_namespace.h.

#### 2.4.2.62 Variable optix::ContextObj::queryVariable ( const std::string & *name* ) const `[inline, virtual]`

Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2132 of file optixpp_namespace.h.

#### 2.4.2.63 void optix::ContextObj::removeVariable ( Variable *v* ) `[inline, virtual]`

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2139 of file optixpp_namespace.h.

#### 2.4.2.64 void optix::ContextObj::setCPUNumThreads ( int *cpu_num_threads* ) `[inline]`

See rtContextSetAttribute

Definition at line 1958 of file optixpp_namespace.h.

#### 2.4.2.65 template<class Iterator > void optix::ContextObj::setDevices ( Iterator *begin*, Iterator *end* ) `[inline]`

See rtContextSetDevices

Definition at line 1892 of file optixpp_namespace.h.

#### 2.4.2.66 void optix::ContextObj::setEntryPointCount ( unsigned int *num_entry_points* ) `[inline]`

See rtContextSetEntryPointCount.

Definition at line 1985 of file optixpp_namespace.h.

#### 2.4.2.67 void optix::ContextObj::setExceptionEnabled ( RTexception *exception,* bool *enabled* ) [inline]

See rtContextSetExceptionEnabled.

Definition at line 2024 of file optixpp_namespace.h.

#### 2.4.2.68 void optix::ContextObj::setExceptionProgram ( unsigned int *entry_point_index,* Program *program* ) [inline]

See rtContextSetExceptionProgram.

Definition at line 2011 of file optixpp_namespace.h.

#### 2.4.2.69 void optix::ContextObj::setGPUPagingForcedOff ( int *gpu_paging_forced_off* ) [inline]

See rtContextSetAttribute.

Definition at line 1963 of file optixpp_namespace.h.

#### 2.4.2.70 void optix::ContextObj::setMissProgram ( unsigned int *ray_type_index,* Program *program* ) [inline]

See rtContextSetMissProgram.

Definition at line 2049 of file optixpp_namespace.h.

#### 2.4.2.71 void optix::ContextObj::setPrintBufferSize ( RTsize *buffer_size_bytes* ) [inline]

See rtContextSetPrintBufferSize.

Definition at line 2101 of file optixpp_namespace.h.

#### 2.4.2.72 void optix::ContextObj::setPrintEnabled ( bool *enabled* ) [inline]

See rtContextSetPrintEnabled

Definition at line 2089 of file optixpp_namespace.h.

#### 2.4.2.73 void optix::ContextObj::setPrintLaunchIndex ( int *x,* int *y* = -1, int *z* = -1 ) [inline]

See rtContextSetPrintLaunchIndex.

Definition at line 2113 of file optixpp_namespace.h.

#### 2.4.2.74 void optix::ContextObj::setRayGenerationProgram ( unsigned int *entry_point_index,* Program *program* ) [inline]

See rtContextSetRayGenerationProgram

Definition at line 1998 of file optixpp_namespace.h.

#### 2.4.2.75 void optix::ContextObj::setRayTypeCount ( unsigned int *num_ray_types* ) [inline]

See rtContextSetRayTypeCount.

Definition at line 2037 of file optixpp_namespace.h.

#### 2.4.2.76 void optix::ContextObj::setStackSize ( RTsize *stack_size_bytes* ) [inline]

See rtContextSetStackSize

Definition at line 1968 of file optixpp_namespace.h.

#### 2.4.2.77 void optix::ContextObj::setTimeoutCallback ( RTtimeoutcallback *callback*, double *min_polling_seconds* ) [inline]

See rtContextSetTimeoutCallback RTtimeoutcallback is defined as typedef int (∗R-Ttimeoutcallback)(void).

Definition at line 1980 of file optixpp_namespace.h.

#### 2.4.2.78 void optix::ContextObj::validate ( ) [inline, virtual]

See rtContextValidate.

Implements optix::DestroyableObj.

Definition at line 1597 of file optixpp_namespace.h.

### 2.4.3 Friends And Related Function Documentation

#### 2.4.3.1 friend class Handle< ContextObj > [friend]

Definition at line 886 of file optixpp_namespace.h.

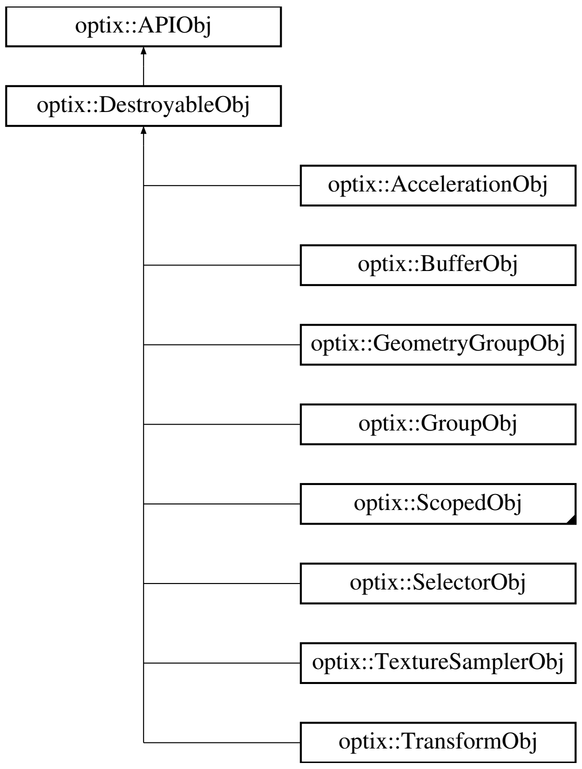
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.5 optix::DestroyableObj Class Reference

```
#include <optixpp_namespace.h>
```

Inheritance diagram for optix::DestroyableObj:



### Public Member Functions

- virtual ~DestroyableObj ()
- virtual void destroy ()=0
- virtual void validate ()=0

### 2.5.1 Detailed Description

Base class for all wrapper objects which can be destroyed and validated.

Wraps:

- RTcontext
- RTgeometry
- RTgeometryinstance
- RTgeometrygroup
- RTgroup
- RTmaterial
- RTprogram

- RTselector
- RTtexturesampler
- RTtransform

Definition at line 341 of file optixpp_namespace.h.

### 2.5.2 Constructor & Destructor Documentation

#### 2.5.2.1 virtual **optix::DestroyableObj::∼DestroyableObj ( )** [inline, virtual]

Definition at line 343 of file optixpp_namespace.h.

### 2.5.3 Member Function Documentation

#### 2.5.3.1 virtual void **optix::DestroyableObj::destroy ( )** [pure virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implemented in optix::BufferObj, optix::TextureSamplerObj, optix::MaterialObj, optix::-GeometryObj, optix::GeometryInstanceObj, optix::AccelerationObj, optix::SelectorObj, optix::TransformObj, optix::GeometryGroupObj, optix::GroupObj, optix::ProgramObj, and optix::ContextObj.

#### 2.5.3.2 virtual void **optix::DestroyableObj::validate ( )** [pure virtual]

call rt[ObjectType]Validate on the underlying OptiX C object

Implemented in optix::BufferObj, optix::TextureSamplerObj, optix::MaterialObj, optix::-GeometryObj, optix::GeometryInstanceObj, optix::AccelerationObj, optix::SelectorObj, optix::TransformObj, optix::GeometryGroupObj, optix::GroupObj, optix::ProgramObj, and optix::ContextObj.

The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.6 optix::Exception Class Reference

```
#include <optixpp_namespace.h>
```

### Public Member Functions

- Exception (const std::string &message, RTresult error_code=RT_ERROR_UNK-NOWN)
- virtual ∼Exception () throw ()
- const std::string & getErrorString () const

- RTresult getErrorCode () const
- virtual const char ∗ what () const throw ()

### Static Public Member Functions

- static Exception makeException (RTresult code, RTcontext context)

### 2.6.1 Detailed Description

Exception class for error reporting from the OptiXpp API.

Encapsulates an error message, often the direct result of a failed OptiX C API function call and subsequent rtContextGetErrorString call.

Definition at line 235 of file optixpp_namespace.h.

### 2.6.2 Constructor & Destructor Documentation

#### 2.6.2.1 optix::Exception::Exception ( const std::string & *message*, RTresult *error_code* = RT_ERROR_UNKNOWN ) [inline]

Create exception.

Definition at line 238 of file optixpp_namespace.h.

#### 2.6.2.2 virtual optix::Exception::∼Exception ( ) throw () [inline, virtual]

Virtual destructor (needed for virtual function calls inherited from std::exception).

Definition at line 243 of file optixpp_namespace.h.

### 2.6.3 Member Function Documentation

#### 2.6.3.1 RTresult optix::Exception::getErrorCode ( ) const [inline]

Retrieve the error code.

Definition at line 249 of file optixpp_namespace.h.

#### 2.6.3.2 const std::string& optix::Exception::getErrorString ( ) const [inline]

Retrieve the error message.

Definition at line 246 of file optixpp_namespace.h.

#### 2.6.3.3 Exception optix::Exception::makeException ( RTresult *code*, RTcontext *context* ) [inline, static]

Helper for creating exceptions from an RTresult code origination from an OptiX C API function call.

Definition at line 262 of file optixpp_namespace.h.

#### 2.6.3.4 virtual const char∗ **optix::Exception::what ( ) const throw ()** `[inline, virtual]`

From std::exception.

Definition at line 256 of file optixpp_namespace.h.

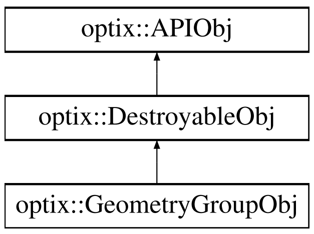
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.7 optix::GeometryGroupObj Class Reference

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::GeometryGroupObj:



### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- RTgeometrygroup get ()

### Friends

- class Handle< GeometryGroupObj >

- void setAcceleration (Acceleration acceleration)
- Acceleration getAcceleration () const

- void setChildCount (unsigned int count)
- unsigned int getChildCount () const
- void setChild (unsigned int index, GeometryInstance geometryinstance)
- GeometryInstance getChild (unsigned int index) const

### 2.7.1 Detailed Description

GeometryGroup wraps the OptiX C API RTgeometrygroup opaque type and its associated function set.

Definition at line 969 of file optixpp_namespace.h.

### 2.7.2 Member Function Documentation

#### 2.7.2.1 void optix::GeometryGroupObj::destroy ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2378 of file optixpp_namespace.h.

#### 2.7.2.2 RTgeometrygroup optix::GeometryGroupObj::get ( ) [inline]

Get the underlying OptiX C API RTgeometrygroup opaque pointer.

Definition at line 2433 of file optixpp_namespace.h.

#### 2.7.2.3 Acceleration optix::GeometryGroupObj::getAcceleration ( ) const [inline]

Query the Acceleration structure for this group. See rtGeometryGroupGetAcceleration.

Definition at line 2402 of file optixpp_namespace.h.

#### 2.7.2.4 GeometryInstance optix::GeometryGroupObj::getChild ( unsigned int *index* ) const [inline]

Query an indexed GeometryInstance within this group. See rtGeometryGroupGetChild.

Definition at line 2426 of file optixpp_namespace.h.

#### 2.7.2.5 unsigned int optix::GeometryGroupObj::getChildCount ( ) const [inline]

Query the number of children for this group. See rtGeometryGroupGetChildCount.

Definition at line 2414 of file optixpp_namespace.h.

#### 2.7.2.6 Context optix::GeometryGroupObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2390 of file optixpp_namespace.h.

#### 2.7.2.7 void optix::GeometryGroupObj::setAcceleration ( Acceleration *acceleration* ) [inline]

Set the Acceleration structure for this group. See rtGeometryGroupSetAcceleration.

Definition at line 2397 of file optixpp_namespace.h.

#### 2.7.2.8 void optix::GeometryGroupObj::setChild ( unsigned int *index*, GeometryInstance *geometryinstance* ) [inline]

Set an indexed GeometryInstance child of this group. See rtGeometryGroupSetChild.

Definition at line 2421 of file optixpp_namespace.h.

#### 2.7.2.9 void optix::GeometryGroupObj::setChildCount ( unsigned int *count* ) [inline]

Set the number of children for this group. See rtGeometryGroupSetChildCount.

Definition at line 2409 of file optixpp_namespace.h.

#### 2.7.2.10 void optix::GeometryGroupObj::validate ( ) [inline, virtual]

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2385 of file optixpp_namespace.h.

### 2.7.3 Friends And Related Function Documentation

#### 2.7.3.1 friend class Handle< GeometryGroupObj > [friend]

Definition at line 1002 of file optixpp_namespace.h.

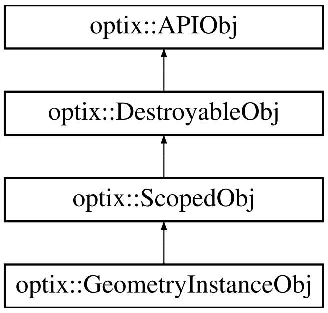
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.8 optix::GeometryInstanceObj Class Reference

```
#include <optixpp_namespace.h>
```

Inheritance diagram for optix::GeometryInstanceObj:



#### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- RTgeometryinstance get ()

#### Friends

- class Handle< GeometryInstanceObj >

- void setGeometry (Geometry geometry)
- Geometry getGeometry () const
- void setMaterialCount (unsigned int count)
- unsigned int getMaterialCount () const
- void setMaterial (unsigned int idx, Material material)
- Material getMaterial (unsigned int idx) const
- unsigned int addMaterial (Material material)

- Variable declareVariable (const std::string &name)
- Variable queryVariable (const std::string &name) const
- void removeVariable (Variable v)
- unsigned int getVariableCount () const
- Variable getVariable (unsigned int index) const

### 2.8.1 Detailed Description

GeometryInstance wraps the OptiX C API RTgeometryinstance acceleration opaque type and its associated function set.

Definition at line 1160 of file optixpp_namespace.h.

### 2.8.2 Member Function Documentation

#### 2.8.2.1 unsigned int **optix::GeometryInstanceObj::addMaterial ( Material *material* )** `[inline]`

Adds the provided material and returns the index to newly added material; increases material count by one.

Definition at line 2631 of file optixpp_namespace.h.

#### 2.8.2.2 **Variable optix::GeometryInstanceObj::declareVariable ( const std::string & *name* )** `[inline, virtual]`

Declare a variable associated with this object. See rt[ObjectType]DeclareVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2639 of file optixpp_namespace.h.

#### 2.8.2.3 void **optix::GeometryInstanceObj::destroy ( )** `[inline, virtual]`

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2575 of file optixpp_namespace.h.

#### 2.8.2.4 RTgeometryinstance **optix::GeometryInstanceObj::get ( )** `[inline]`

Get the underlying OptiX C API RTgeometryinstance opaque pointer.

Definition at line 2672 of file optixpp_namespace.h.

#### 2.8.2.5 **Context optix::GeometryInstanceObj::getContext ( ) const** `[inline, virtual]`

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2587 of file optixpp_namespace.h.

#### 2.8.2.6 **Geometry optix::GeometryInstanceObj::getGeometry ( ) const** `[inline]`

Get the geometry object associated with this instance. See rtGeometryInstanceGet-Geometry.

Definition at line 2599 of file optixpp_namespace.h.

#### 2.8.2.7 **Material optix::GeometryInstanceObj::getMaterial ( unsigned int *idx* ) const** `[inline]`

Get the material at given index. See rtGeometryInstanceGetMaterial.

Definition at line 2623 of file optixpp_namespace.h.

#### 2.8.2.8 unsigned int optix::GeometryInstanceObj::getMaterialCount ( ) const [inline]

Query the number of materials associated with this instance. See rtGeometryInstance-GetMaterialCount.

Definition at line 2611 of file optixpp_namespace.h.

#### 2.8.2.9 Variable optix::GeometryInstanceObj::getVariable ( unsigned int *index* ) const [inline, virtual]

Query variable by index. See rt[ObjectType]GetVariable.

Implements optix::ScopedObj.

Definition at line 2665 of file optixpp_namespace.h.

#### 2.8.2.10 unsigned int optix::GeometryInstanceObj::getVariableCount ( ) const [inline, virtual]

Query the number of variables associated with this object. Used along with ScopedObj-::getVariable to iterate over variables in an object. See rt[ObjectType]GetVariableCount

Implements optix::ScopedObj.

Definition at line 2658 of file optixpp_namespace.h.

#### 2.8.2.11 Variable optix::GeometryInstanceObj::queryVariable ( const std::string & *name* ) const [inline, virtual]

Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2646 of file optixpp_namespace.h.

#### 2.8.2.12 void optix::GeometryInstanceObj::removeVariable ( Variable *v* ) [inline, virtual]

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2653 of file optixpp_namespace.h.

#### 2.8.2.13 void optix::GeometryInstanceObj::setGeometry ( Geometry *geometry* ) [inline]

Set the geometry object associated with this instance. See rtGeometryInstanceSet-Geometry.

Definition at line 2594 of file optixpp_namespace.h.

#### 2.8.2.14 void optix::GeometryInstanceObj::setMaterial ( unsigned int *idx*, Material *material* ) [inline]

Set the material at given index. See rtGeometryInstanceSetMaterial.

Definition at line 2618 of file optixpp_namespace.h.

#### 2.8.2.15 void optix::GeometryInstanceObj::setMaterialCount ( unsigned int *count* ) [inline]

Set the number of materials associated with this instance. See rtGeometryInstanceSet-MaterialCount.

Definition at line 2606 of file optixpp_namespace.h.

#### 2.8.2.16 void optix::GeometryInstanceObj::validate ( ) [inline, virtual]

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2582 of file optixpp_namespace.h.

### 2.8.3 Friends And Related Function Documentation

#### 2.8.3.1 friend class Handle< GeometryInstanceObj > [friend]

Definition at line 1202 of file optixpp_namespace.h.

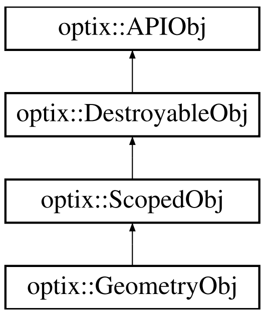
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.9 optix::GeometryObj Class Reference

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::GeometryObj:



### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- RTgeometry get ()

### Friends

- class Handle< GeometryObj >

- void markDirty ()
- bool isDirty () const

- void setPrimitiveCount (unsigned int num_primitives)
- unsigned int getPrimitiveCount () const

- void setBoundingBoxProgram (Program program)
- Program getBoundingBoxProgram () const
- void setIntersectionProgram (Program program)
- Program getIntersectionProgram () const

- Variable declareVariable (const std::string &name)
- Variable queryVariable (const std::string &name) const
- void removeVariable (Variable v)
- unsigned int getVariableCount () const
- Variable getVariable (unsigned int index) const

### 2.9.1 Detailed Description

Geometry wraps the OptiX C API RTgeometry opaque type and its associated function set.

Definition at line 1212 of file optixpp_namespace.h.

### 2.9.2 Member Function Documentation

#### 2.9.2.1 Variable optix::GeometryObj::declareVariable ( const std::string & *name* ) [inline, virtual]

Declare a variable associated with this object. See rt[ObjectType]DeclareVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2732 of file optixpp_namespace.h.

#### 2.9.2.2 void optix::GeometryObj::destroy ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2677 of file optixpp_namespace.h.

#### 2.9.2.3 RTgeometry optix::GeometryObj::get ( ) [inline]

Get the underlying OptiX C API RTgeometry opaque pointer.

Definition at line 2777 of file optixpp_namespace.h.

#### 2.9.2.4 Program optix::GeometryObj::getBoundingBoxProgram ( ) const [inline]

Get the bounding box program for this geometry. See rtGeometryGetBoundingBox-Program.

Definition at line 2713 of file optixpp_namespace.h.

#### 2.9.2.5 Context optix::GeometryObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2689 of file optixpp_namespace.h.

#### 2.9.2.6 Program optix::GeometryObj::getIntersectionProgram ( ) const [inline]

Get the intersection program for this geometry. See rtGeometryGetIntersection-Program.

Definition at line 2725 of file optixpp_namespace.h.

#### 2.9.2.7 unsigned int optix::GeometryObj::getPrimitiveCount ( ) const [inline]

Query the number of primitives in this geometry objects (eg, number of triangles in mesh). See rtGeometryGetPrimitiveCount

Definition at line 2701 of file optixpp_namespace.h.

#### 2.9.2.8 Variable optix::GeometryObj::getVariable ( unsigned int *index* ) const [inline, virtual]

Query variable by index. See rt[ObjectType]GetVariable.

Implements optix::ScopedObj.

Definition at line 2758 of file optixpp_namespace.h.

#### 2.9.2.9 unsigned int optix::GeometryObj::getVariableCount ( ) const [inline, virtual]

Query the number of variables associated with this object. Used along with ScopedObj::getVariable to iterate over variables in an object. See rt[ObjectType]GetVariableCount

Implements optix::ScopedObj.

Definition at line 2751 of file optixpp_namespace.h.

#### 2.9.2.10 bool optix::GeometryObj::isDirty ( ) const [inline]

Query whether this geometry has been marked dirty. See rtGeometryIsDirty.

Definition at line 2770 of file optixpp_namespace.h.

#### 2.9.2.11 void optix::GeometryObj::markDirty ( ) [inline]

Mark this geometry as dirty, causing rebuild of parent groups acceleration. See rtGeometryMarkDirty.

Definition at line 2765 of file optixpp_namespace.h.

#### 2.9.2.12 Variable optix::GeometryObj::queryVariable ( const std::string & *name* ) const [inline, virtual]

Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2739 of file optixpp_namespace.h.

#### 2.9.2.13 void optix::GeometryObj::removeVariable ( Variable *v* ) [inline, virtual]

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2746 of file optixpp_namespace.h.

#### 2.9.2.14 void optix::GeometryObj::setBoundingBoxProgram ( Program *program* ) [inline]

Set the bounding box program for this geometry. See rtGeometrySetBoundingBoxProgram.

Definition at line 2708 of file optixpp_namespace.h.

#### 2.9.2.15 void optix::GeometryObj::setIntersectionProgram ( Program *program* ) [inline]

Set the intersection program for this geometry. See rtGeometrySetIntersectionProgram.

Definition at line 2720 of file optixpp_namespace.h.

#### 2.9.2.16 void optix::GeometryObj::setPrimitiveCount ( unsigned int *num_primitives* ) [inline]

Set the number of primitives in this geometry objects (eg, number of triangles in mesh). See rtGeometrySetPrimitiveCount

Definition at line 2696 of file optixpp_namespace.h.

#### 2.9.2.17 void optix::GeometryObj::validate ( ) [inline, virtual]

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2684 of file optixpp_namespace.h.

### 2.9.3 Friends And Related Function Documentation

#### 2.9.3.1 friend class Handle< GeometryObj > [friend]

Definition at line 1262 of file optixpp_namespace.h.

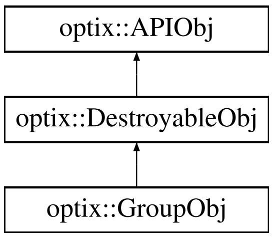
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.10 optix::GroupObj Class Reference

```
#include <optixpp_namespace.h>
```

Inheritance diagram for optix::GroupObj:



### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- RTgroup get ()

#### Friends

- class Handle< GroupObj >

- void setAcceleration (Acceleration acceleration)
- Acceleration getAcceleration () const

- void setChildCount (unsigned int count)
- unsigned int getChildCount () const
- template<typename T >
  void setChild (unsigned int index, T child)
- template<typename T >
  T getChild (unsigned int index) const

### 2.10.1 Detailed Description

Group wraps the OptiX C API RTgroup opaque type and its associated function set.

Definition at line 925 of file optixpp_namespace.h.

### 2.10.2 Member Function Documentation

#### 2.10.2.1 void **optix::GroupObj::destroy** ( ) `[inline, virtual]`

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2221 of file optixpp_namespace.h.

#### 2.10.2.2 RTgroup **optix::GroupObj::get** ( ) `[inline]`

Get the underlying OptiX C API RTgroup opaque pointer.

Definition at line 2373 of file optixpp_namespace.h.

#### 2.10.2.3 **Acceleration optix::GroupObj::getAcceleration** ( ) const `[inline]`

Query the Acceleration structure for this group. See rtGroupGetAcceleration.

Definition at line 2340 of file optixpp_namespace.h.

#### 2.10.2.4 template<typename T > T **optix::GroupObj::getChild** ( unsigned int *index* ) const `[inline]`

Query an indexed child within this group. See rtGroupGetChild.

Definition at line 2366 of file optixpp_namespace.h.

#### 2.10.2.5 unsigned int **optix::GroupObj::getChildCount** ( ) const `[inline]`

Query the number of children for this group. See rtGroupGetChildCount.

Definition at line 2352 of file optixpp_namespace.h.

#### 2.10.2.6 Context optix::GroupObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2233 of file optixpp_namespace.h.

#### 2.10.2.7 void optix::GroupObj::setAcceleration ( Acceleration *acceleration* ) [inline]

Set the Acceleration structure for this group. See rtGroupSetAcceleration.

Definition at line 2335 of file optixpp_namespace.h.

#### 2.10.2.8 template<typename T > void optix::GroupObj::setChild ( unsigned int *index*, T *child* ) [inline]

Set an indexed child within this group. See rtGroupSetChild.

Definition at line 2360 of file optixpp_namespace.h.

#### 2.10.2.9 void optix::GroupObj::setChildCount ( unsigned int *count* ) [inline]

Set the number of children for this group. See rtGroupSetChildCount.

Definition at line 2347 of file optixpp_namespace.h.

#### 2.10.2.10 void optix::GroupObj::validate ( ) [inline, virtual]

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2228 of file optixpp_namespace.h.

### 2.10.3 Friends And Related Function Documentation

#### 2.10.3.1 friend class Handle< GroupObj > [friend]

Definition at line 959 of file optixpp_namespace.h.

The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.11 optix::Handle< T > Class Template Reference

```
#include <optixpp_namespace.h>
```

### Public Member Functions

- Handle ()
- Handle (T ∗ptr)
- template<class U >
  Handle (U ∗ptr)
- Handle (const Handle< T > &copy)
- template<class U >
  Handle (const Handle< U > &copy)
- Handle< T > & operator= (const Handle< T > &copy)
- template<class U >
  Handle< T > & operator= (const Handle< U > &copy)
- ∼Handle ()
- T ∗ operator-> ()
- const T ∗ operator-> () const
- T ∗ get ()
- const T ∗ get () const
- operator bool () const
- Handle< VariableObj > operator[] (const std::string &varname)
- Handle< VariableObj > operator[] (const char ∗varname)

### Static Public Member Functions

- static Handle< T > take (typename T::api_t p)
- static Handle< T > take (RTobject p)
- static Handle< T > create ()
- static unsigned int getDeviceCount ()

### 2.11.1 Detailed Description

**template<class T>class optix::Handle< T >**

The Handle class is a reference counted handle class used to manipulate API objects.

All interaction with API objects should be done via these handles and the associated typedefs rather than direct usage of the objects.

Definition at line 123 of file optixpp_namespace.h.

### 2.11.2 Constructor & Destructor Documentation

#### 2.11.2.1 template<class T> optix::Handle< T >::Handle ( ) [inline]

Default constructor initializes handle to null pointer.

Definition at line 126 of file optixpp_namespace.h.

#### 2.11.2.2 template<class T> optix::Handle< T >::Handle ( T ∗ *ptr* ) `[inline]`

Takes a raw pointer to an API object and creates a handle.

Definition at line 129 of file optixpp_namespace.h.

#### 2.11.2.3 template<class T> template<class U > optix::Handle< T >::Handle ( U ∗ *ptr* ) `[inline]`

Takes a raw pointer of arbitrary type and creates a handle.

Definition at line 133 of file optixpp_namespace.h.

#### 2.11.2.4 template<class T> optix::Handle< T >::Handle ( const Handle< T > & *copy* ) `[inline]`

Takes a handle of the same type and creates a handle.

Definition at line 136 of file optixpp_namespace.h.

#### 2.11.2.5 template<class T> template<class U > optix::Handle< T >::Handle ( const Handle< U > & *copy* ) `[inline]`

Takes a handle of some other type and creates a handle.

Definition at line 140 of file optixpp_namespace.h.

#### 2.11.2.6 template<class T> optix::Handle< T >::~Handle ( ) `[inline]`

Decrements reference count on the handled object.

Definition at line 152 of file optixpp_namespace.h.

### 2.11.3 Member Function Documentation

#### 2.11.3.1 template<class T> static Handle<T> optix::Handle< T >::create ( ) `[inline, static]`

Static object creation. Only valid for contexts.

Definition at line 197 of file optixpp_namespace.h.

#### 2.11.3.2 template<class T> T∗ optix::Handle< T >::get ( ) `[inline]`

Retrieve the handled object.

Definition at line 165 of file optixpp_namespace.h.

#### 2.11.3.3 template<class T> const T∗ optix::Handle< T >::get ( ) const `[inline]`

Definition at line 166 of file optixpp_namespace.h.

#### 2.11.3.4 template<class T> static unsigned int optix::Handle< T >::getDeviceCount ( ) [inline, static]

Query the machine device count. Only valid for contexts.

Definition at line 200 of file optixpp_namespace.h.

#### 2.11.3.5 template<class T> optix::Handle< T >::operator bool ( ) const [inline]

implicit bool cast based on NULLness of wrapped pointer

Definition at line 169 of file optixpp_namespace.h.

#### 2.11.3.6 template<class T> T∗ optix::Handle< T >::operator-> ( ) [inline]

Dereferences the handle.

Definition at line 161 of file optixpp_namespace.h.

#### 2.11.3.7 template<class T> const T∗ optix::Handle< T >::operator-> ( ) const [inline]

Definition at line 162 of file optixpp_namespace.h.

#### 2.11.3.8 template<class T> Handle<T>& optix::Handle< T >::operator= ( const Handle< T > & *copy* ) [inline]

Assignment of handle with same underlying object type.

Definition at line 143 of file optixpp_namespace.h.

#### 2.11.3.9 template<class T> template<class U > Handle<T>& optix::Handle< T >::operator= ( const Handle< U > & *copy* ) [inline]

Assignment of handle with different underlying object type.

Definition at line 148 of file optixpp_namespace.h.

#### 2.11.3.10 template<class T > Handle< VariableObj > optix::Handle< T >::operator[] ( const std::string & *varname* )

Variable access operator. This operator will query the API object for a variable with the given name, creating a new variable instance if necessary. Only valid for ScopedObjs.

Definition at line 598 of file optixpp_namespace.h.

#### 2.11.3.11 template<class T > Handle< VariableObj > optix::Handle< T >::operator[] ( const char ∗ *varname* )

Variable access operator. Identical to operator[](const std::string& varname)

Explicitly define char∗ version to avoid ambiguities between builtin operator[](int, char∗) and Handle::operator[]( std::string ). The problem lies in that a Handle can be cast to a bool then to an int which implies that:

```
Context context;
```

```
context["var"];
```

can be interpreted as either

```
1["var"]; // Strange but legal way to index into a string (same as "var"[1]
   )
```

or

```
context[ std::string("var") ];
```

Definition at line 607 of file optixpp_namespace.h.

#### 2.11.3.12 template<class T> static Handle<T> optix::Handle< T >::take ( typename T::api_t *p* ) [inline, static]

Takes a base optix api opaque type and creates a handle to optixpp wrapper type.

Definition at line 155 of file optixpp_namespace.h.

#### 2.11.3.13 template<class T> static Handle<T> optix::Handle< T >::take ( RTobject *p* ) [inline, static]

Special version that takes an RTobject which must be cast up to the appropriate OptiX API opaque type.

Definition at line 158 of file optixpp_namespace.h.

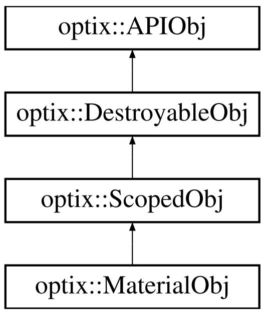
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.12 optix::MaterialObj Class Reference

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::MaterialObj:



#### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- RTmaterial get ()

#### Friends

- class Handle< MaterialObj >

- void setClosestHitProgram (unsigned int ray_type_index, Program program)
- Program getClosestHitProgram (unsigned int ray_type_index) const
- void setAnyHitProgram (unsigned int ray_type_index, Program program)
- Program getAnyHitProgram (unsigned int ray_type_index) const

- Variable declareVariable (const std::string &name)
- Variable queryVariable (const std::string &name) const
- void removeVariable (Variable v)
- unsigned int getVariableCount () const
- Variable getVariable (unsigned int index) const

### 2.12.1 Detailed Description

Material wraps the OptiX C API RTmaterial opaque type and its associated function set.

Definition at line 1272 of file optixpp_namespace.h.

### 2.12.2 Member Function Documentation

#### 2.12.2.1 Variable optix::MaterialObj::declareVariable ( const std::string & *name* ) `[inline, virtual]`

Declare a variable associated with this object. See rt[ObjectType]DeclareVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2825 of file optixpp_namespace.h.

#### 2.12.2.2 void optix::MaterialObj::destroy ( ) `[inline, virtual]`

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2782 of file optixpp_namespace.h.

#### 2.12.2.3 RTmaterial optix::MaterialObj::get ( ) [inline]

Get the underlying OptiX C API RTmaterial opaque pointer.

Definition at line 2858 of file optixpp_namespace.h.

#### 2.12.2.4 Program optix::MaterialObj::getAnyHitProgram ( unsigned int *ray_type_index* ) const [inline]

Get any hit program for this material at the given *ray_type* index. See rtMaterialGetAny-HitProgram.

Definition at line 2818 of file optixpp_namespace.h.

#### 2.12.2.5 Program optix::MaterialObj::getClosestHitProgram ( unsigned int *ray_type_index* ) const [inline]

Get closest hit program for this material at the given *ray_type* index. See rtMaterialGet-ClosestHitProgram.

Definition at line 2806 of file optixpp_namespace.h.

#### 2.12.2.6 Context optix::MaterialObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2794 of file optixpp_namespace.h.

#### 2.12.2.7 Variable optix::MaterialObj::getVariable ( unsigned int *index* ) const [inline, virtual]

Query variable by index. See rt[ObjectType]GetVariable.

Implements optix::ScopedObj.

Definition at line 2851 of file optixpp_namespace.h.

#### 2.12.2.8 unsigned int optix::MaterialObj::getVariableCount ( ) const [inline, virtual]

Query the number of variables associated with this object. Used along with ScopedObj-::getVariable to iterate over variables in an object. See rt[ObjectType]GetVariableCount

Implements optix::ScopedObj.

Definition at line 2844 of file optixpp_namespace.h.

#### 2.12.2.9 Variable optix::MaterialObj::queryVariable ( const std::string & *name* ) const [inline, virtual]

Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2832 of file optixpp_namespace.h.

#### 2.12.2.10 void **optix::MaterialObj::removeVariable** ( **Variable** *v* ) `[inline, virtual]`

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2839 of file optixpp_namespace.h.

#### 2.12.2.11 void **optix::MaterialObj::setAnyHitProgram** ( unsigned int *ray_type_index*, **Program** *program* ) `[inline]`

Set any hit program for this material at the given *ray_type* index. See rtMaterialSetAny-HitProgram.

Definition at line 2813 of file optixpp_namespace.h.

#### 2.12.2.12 void **optix::MaterialObj::setClosestHitProgram** ( unsigned int *ray_type_index*, **Program** *program* ) `[inline]`

Set closest hit program for this material at the given *ray_type* index. See rtMaterialSet-ClosestHitProgram.

Definition at line 2801 of file optixpp_namespace.h.

#### 2.12.2.13 void **optix::MaterialObj::validate** ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2789 of file optixpp_namespace.h.

### 2.12.3 Friends And Related Function Documentation

#### 2.12.3.1 friend class **Handle< MaterialObj >** `[friend]`

Definition at line 1305 of file optixpp_namespace.h.

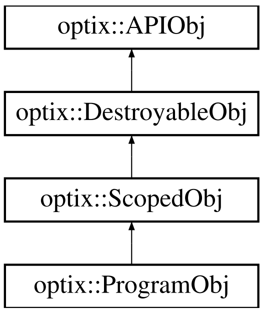
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.13 optix::ProgramObj Class Reference

```
#include <optixpp_namespace.h>
```

Inheritance diagram for optix::ProgramObj:



### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- Variable declareVariable (const std::string &name)
- Variable queryVariable (const std::string &name) const
- void removeVariable (Variable v)
- unsigned int getVariableCount () const
- Variable getVariable (unsigned int index) const
- RTprogram get ()

### Friends

- class Handle< ProgramObj >

### 2.13.1 Detailed Description

Program object wraps the OptiX C API RTprogram opaque type and its associated function set.

Definition at line 896 of file optixpp_namespace.h.

### 2.13.2 Member Function Documentation

#### 2.13.2.1 Variable optix::ProgramObj::declareVariable ( const std::string & *name* ) `[inline, virtual]`

Declare a variable associated with this object. See rt[ObjectType]DeclareVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2183 of file optixpp_namespace.h.

#### 2.13.2.2 void optix::ProgramObj::destroy ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2164 of file optixpp_namespace.h.

#### 2.13.2.3 RTprogram optix::ProgramObj::get ( ) [inline]

Definition at line 2216 of file optixpp_namespace.h.

#### 2.13.2.4 Context optix::ProgramObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2176 of file optixpp_namespace.h.

#### 2.13.2.5 Variable optix::ProgramObj::getVariable ( unsigned int *index* ) const [inline, virtual]

Query variable by index. See rt[ObjectType]GetVariable.

Implements optix::ScopedObj.

Definition at line 2209 of file optixpp_namespace.h.

#### 2.13.2.6 unsigned int optix::ProgramObj::getVariableCount ( ) const [inline, virtual]

Query the number of variables associated with this object. Used along with ScopedObj::getVariable to iterate over variables in an object. See rt[ObjectType]GetVariableCount

Implements optix::ScopedObj.

Definition at line 2202 of file optixpp_namespace.h.

#### 2.13.2.7 Variable optix::ProgramObj::queryVariable ( const std::string & *name* ) const [inline, virtual]

Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implements optix::ScopedObj.

Definition at line 2190 of file optixpp_namespace.h.

#### 2.13.2.8 void optix::ProgramObj::removeVariable ( Variable *v* ) [inline, virtual]

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2197 of file optixpp_namespace.h.

#### 2.13.2.9 void optix::ProgramObj::validate ( ) [inline, virtual]

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2171 of file optixpp_namespace.h.

### 2.13.3 Friends And Related Function Documentation

#### 2.13.3.1 friend class Handle< ProgramObj > [friend]

Definition at line 915 of file optixpp_namespace.h.

The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.14 RTUtraversalresult Struct Reference

```
#include <optixu_traversal.h>
```

**Public Attributes**

- int prim_id
- float t

### 2.14.1 Detailed Description

Structure encapsulating the result of a single ray query.

Definition at line 35 of file optixu_traversal.h.

### 2.14.2 Member Data Documentation

#### 2.14.2.1 int RTUtraversalresult::prim_id

Index of the interesected triangle, -1 for miss

Definition at line 36 of file optixu_traversal.h.

#### 2.14.2.2 float RTUtraversalresult::t

Ray t parameter of hit point

Definition at line 37 of file optixu_traversal.h.

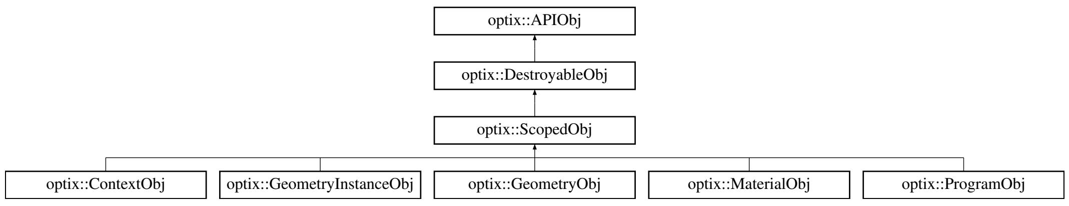
The documentation for this struct was generated from the following file:

- optixu_traversal.h

## 2.15 optix::ScopedObj Class Reference

```
#include <optixpp_namespace.h>
```

Inheritance diagram for optix::ScopedObj:



### Public Member Functions

- virtual ∼ScopedObj ()
- virtual Variable declareVariable (const std::string &name)=0
- virtual Variable queryVariable (const std::string &name) const =0
- virtual void removeVariable (Variable v)=0
- virtual unsigned int getVariableCount () const =0
- virtual Variable getVariable (unsigned int index) const =0

### 2.15.1 Detailed Description

Base class for all objects which are OptiX variable containers.

Wraps:

- RTcontext
- RTgeometry
- RTgeometryinstance
- RTmaterial
- RTprogram

Definition at line 367 of file optixpp_namespace.h.

### 2.15.2 Constructor & Destructor Documentation

#### 2.15.2.1 virtual **optix::ScopedObj::∼ScopedObj ( )** `[inline, virtual]`

Definition at line 369 of file optixpp_namespace.h.

### 2.15.3 Member Function Documentation

#### 2.15.3.1 virtual **Variable optix::ScopedObj::declareVariable (** const std::string & *name* **)** `[pure virtual]`

Declare a variable associated with this object. See rt[ObjectType]DeclareVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implemented in optix::MaterialObj, optix::GeometryObj, optix::GeometryInstanceObj, optix::ProgramObj, and optix::ContextObj.

#### 2.15.3.2 virtual **Variable optix::ScopedObj::getVariable (** unsigned int *index* **)** const `[pure virtual]`

Query variable by index. See rt[ObjectType]GetVariable.

Implemented in optix::MaterialObj, optix::GeometryObj, optix::GeometryInstanceObj, optix::ProgramObj, and optix::ContextObj.

#### 2.15.3.3 virtual unsigned int **optix::ScopedObj::getVariableCount (** **)** const `[pure virtual]`

Query the number of variables associated with this object. Used along with ScopedObj::getVariable to iterate over variables in an object. See rt[ObjectType]GetVariableCount

Implemented in optix::MaterialObj, optix::GeometryObj, optix::GeometryInstanceObj, optix::ProgramObj, and optix::ContextObj.

#### 2.15.3.4 virtual **Variable optix::ScopedObj::queryVariable (** const std::string & *name* **)** const `[pure virtual]`

Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function Handle::operator[].

Implemented in optix::MaterialObj, optix::GeometryObj, optix::GeometryInstanceObj, optix::ProgramObj, and optix::ContextObj.

#### 2.15.3.5 virtual void **optix::ScopedObj::removeVariable (** **Variable** *v* **)** `[pure virtual]`

Remove a variable associated with this object.

Implemented in optix::MaterialObj, optix::GeometryObj, optix::GeometryInstanceObj, optix::ProgramObj, and optix::ContextObj.

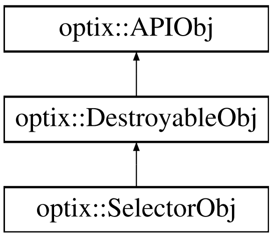
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.16 optix::SelectorObj Class Reference

```
#include <optixpp_namespace.h>
```

Inheritance diagram for optix::SelectorObj:



### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- RTselector get ()

### Friends

- class Handle< SelectorObj >

- void setVisitProgram (Program program)
- Program getVisitProgram () const

- void setChildCount (unsigned int count)
- unsigned int getChildCount () const
- template<typename T >
  void setChild (unsigned int index, T child)
- template<typename T >
  T getChild (unsigned int index) const

- Variable declareVariable (const std::string &name)
- Variable queryVariable (const std::string &name) const
- void removeVariable (Variable v)
- unsigned int getVariableCount () const
- Variable getVariable (unsigned int index) const

### 2.16.1 Detailed Description

Selector wraps the OptiX C API RTselector opaque type and its associated function set.

Definition at line 1050 of file optixpp_namespace.h.

### 2.16.2 Member Function Documentation

#### 2.16.2.1 Variable optix::SelectorObj::declareVariable ( const std::string & *name* ) `[inline]`

Definition at line 2297 of file optixpp_namespace.h.

#### 2.16.2.2 void optix::SelectorObj::destroy ( ) [inline, virtual]

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2240 of file optixpp_namespace.h.

#### 2.16.2.3 RTselector optix::SelectorObj::get ( ) [inline]

Get the underlying OptiX C API RTselector opaque pointer.

Definition at line 2330 of file optixpp_namespace.h.

#### 2.16.2.4 template<typename T > T optix::SelectorObj::getChild ( unsigned int *index* ) const [inline]

Query an indexed child within this group. See rtSelectorGetChild.

Definition at line 2290 of file optixpp_namespace.h.

#### 2.16.2.5 unsigned int optix::SelectorObj::getChildCount ( ) const [inline]

Query the number of children for this group. See rtSelectorGetChildCount.

Definition at line 2276 of file optixpp_namespace.h.

#### 2.16.2.6 Context optix::SelectorObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2252 of file optixpp_namespace.h.

#### 2.16.2.7 Variable optix::SelectorObj::getVariable ( unsigned int *index* ) const [inline]

Definition at line 2323 of file optixpp_namespace.h.

#### 2.16.2.8 unsigned int optix::SelectorObj::getVariableCount ( ) const [inline]

Definition at line 2316 of file optixpp_namespace.h.

#### 2.16.2.9 Program optix::SelectorObj::getVisitProgram ( ) const [inline]

Get the visitor program for this selector. See rtSelectorGetVisitProgram.

Definition at line 2264 of file optixpp_namespace.h.

#### 2.16.2.10 Variable optix::SelectorObj::queryVariable ( const std::string & *name* ) const [inline]

Definition at line 2304 of file optixpp_namespace.h.

#### 2.16.2.11 void optix::SelectorObj::removeVariable ( Variable *v* ) [inline]

Definition at line 2311 of file optixpp_namespace.h.

#### 2.16.2.12 template<typename T > void optix::SelectorObj::setChild ( unsigned int *index*, T *child* ) [inline]

Set an indexed child child of this group. See rtSelectorSetChild.

Definition at line 2284 of file optixpp_namespace.h.

#### 2.16.2.13 void optix::SelectorObj::setChildCount ( unsigned int *count* ) [inline]

Set the number of children for this group. See rtSelectorSetChildCount.

Definition at line 2271 of file optixpp_namespace.h.

#### 2.16.2.14 void optix::SelectorObj::setVisitProgram ( Program *program* ) [inline]

Set the visitor program for this selector. See rtSelectorSetVisitProgram

Definition at line 2259 of file optixpp_namespace.h.

#### 2.16.2.15 void optix::SelectorObj::validate ( ) [inline, virtual]

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2247 of file optixpp_namespace.h.

### 2.16.3 Friends And Related Function Documentation

#### 2.16.3.1 friend class Handle< SelectorObj > [friend]

Definition at line 1091 of file optixpp_namespace.h.

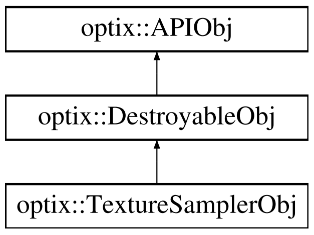
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.17 optix::TextureSamplerObj Class Reference

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::TextureSamplerObj:



### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- RTtexturesampler get ()

### Friends

- class Handle< TextureSamplerObj >

- void setMipLevelCount (unsigned int num_mip_levels)
- unsigned int getMipLevelCount () const
- void setArraySize (unsigned int num_textures_in_array)
- unsigned int getArraySize () const
- void setWrapMode (unsigned int dim, RTwrapmode wrapmode)
- RTwrapmode getWrapMode (unsigned int dim) const
- void setFilteringModes (RTfiltermode minification, RTfiltermode magnification, R-Tfiltermode mipmapping)
- void getFilteringModes (RTfiltermode &minification, RTfiltermode &magnification, RTfiltermode &mipmapping) const
- void setMaxAnisotropy (float value)
- float getMaxAnisotropy () const
- void setReadMode (RTtexturereadmode readmode)
- RTtexturereadmode getReadMode () const
- void setIndexingMode (RTtextureindexmode indexmode)
- RTtextureindexmode getIndexingMode () const

- int getId () const

- void setBuffer (unsigned int texture_array_idx, unsigned int mip_level, Buffer buffer)
- Buffer getBuffer (unsigned int texture_array_idx, unsigned int mip_level) const

- void registerGLTexture ()
- void unregisterGLTexture ()

### 2.17.1 Detailed Description

TextureSampler wraps the OptiX C API RTtexturesampler opaque type and its associated function set.

Definition at line 1315 of file optixpp_namespace.h.

### 2.17.2 Member Function Documentation

#### 2.17.2.1 void **optix::TextureSamplerObj::destroy ( )** `[inline, virtual]`

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2863 of file optixpp_namespace.h.

#### 2.17.2.2 RTtexturesampler **optix::TextureSamplerObj::get ( )** `[inline]`

Get the underlying OptiX C API RTtexturesampler opaque pointer.

Definition at line 2983 of file optixpp_namespace.h.

#### 2.17.2.3 unsigned int **optix::TextureSamplerObj::getArraySize ( )** const `[inline]`

Query the texture array size for this sampler. See rtTextureSamplerGetArraySize.

Definition at line 2899 of file optixpp_namespace.h.

#### 2.17.2.4 **Buffer optix::TextureSamplerObj::getBuffer (** unsigned int *texture_array_idx*, unsigned int *mip_level* ) const `[inline]`

Get the underlying buffer used for texture storage. rtTextureSamplerGetBuffer.

Definition at line 2976 of file optixpp_namespace.h.

#### 2.17.2.5 **Context optix::TextureSamplerObj::getContext ( )** const `[inline, virtual]`

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2875 of file optixpp_namespace.h.

#### 2.17.2.6 void **optix::TextureSamplerObj::getFilteringModes (** RTfiltermode & *minification*, RTfiltermode & *magnification*, RTfiltermode & *mipmapping* ) const `[inline]`

Query filtering modes for this sampler. See rtTextureSamplerGetFilteringModes.

Definition at line 2923 of file optixpp_namespace.h.

#### 2.17.2.7 int optix::TextureSamplerObj::getId ( ) const [inline]

Returns the device-side ID of this sampler.

Definition at line 2940 of file optixpp_namespace.h.

#### 2.17.2.8 RTtextureindexmode optix::TextureSamplerObj::getIndexingMode ( ) const [inline]

Query texture indexing mode for this sampler. See rtTextureSamplerGetIndexingMode.

Definition at line 2964 of file optixpp_namespace.h.

#### 2.17.2.9 float optix::TextureSamplerObj::getMaxAnisotropy ( ) const [inline]

Query maximum anisotropy for this sampler. See rtTextureSamplerGetMaxAnisotropy.

Definition at line 2933 of file optixpp_namespace.h.

#### 2.17.2.10 unsigned int optix::TextureSamplerObj::getMipLevelCount ( ) const [inline]

Query the number of mip levels for this sampler. See rtTextureSamplerGetMipLevel-Count.

Definition at line 2887 of file optixpp_namespace.h.

#### 2.17.2.11 RTtexturereadmode optix::TextureSamplerObj::getReadMode ( ) const [inline]

Query texture read mode for this sampler. See rtTextureSamplerGetReadMode.

Definition at line 2952 of file optixpp_namespace.h.

#### 2.17.2.12 RTwrapmode optix::TextureSamplerObj::getWrapMode ( unsigned int *dim* ) const [inline]

Query the texture wrap mode for this sampler. See rtTextureSamplerGetWrapMode.

Definition at line 2911 of file optixpp_namespace.h.

#### 2.17.2.13 void optix::TextureSamplerObj::registerGLTexture ( ) [inline]

Declare the texture's buffer as mutable and inaccessible by OptiX. See rtTexture-SamplerGLRegister.

Definition at line 2988 of file optixpp_namespace.h.

#### 2.17.2.14 void optix::TextureSamplerObj::setArraySize ( unsigned int *num_textures_in_array* ) [inline]

Set the texture array size for this sampler. See rtTextureSamplerSetArraySize.

Definition at line 2894 of file optixpp_namespace.h.

#### 2.17.2.15 void optix::TextureSamplerObj::setBuffer ( unsigned int *texture_array_idx*, unsigned int *mip_level*, Buffer *buffer* ) [inline]

Set the underlying buffer used for texture storage. rtTextureSamplerSetBuffer.

Definition at line 2971 of file optixpp_namespace.h.

#### 2.17.2.16 void optix::TextureSamplerObj::setFilteringModes ( RTfiltermode *minification*, RTfiltermode *magnification*, RTfiltermode *mipmapping* ) [inline]

Set filtering modes for this sampler. See rtTextureSamplerSetFilteringModes.

Definition at line 2918 of file optixpp_namespace.h.

#### 2.17.2.17 void optix::TextureSamplerObj::setIndexingMode ( RTtextureindexmode *indexmode* ) [inline]

Set texture indexing mode for this sampler. See rtTextureSamplerSetIndexingMode.

Definition at line 2959 of file optixpp_namespace.h.

#### 2.17.2.18 void optix::TextureSamplerObj::setMaxAnisotropy ( float *value* ) [inline]

Set maximum anisotropy for this sampler. See rtTextureSamplerSetMaxAnisotropy.

Definition at line 2928 of file optixpp_namespace.h.

#### 2.17.2.19 void optix::TextureSamplerObj::setMipLevelCount ( unsigned int *num_mip_levels* ) [inline]

Set the number of mip levels for this sampler. See rtTextureSamplerSetMipLevelCount.

Definition at line 2882 of file optixpp_namespace.h.

#### 2.17.2.20 void optix::TextureSamplerObj::setReadMode ( RTtexturereadmode *readmode* ) [inline]

Set texture read mode for this sampler. See rtTextureSamplerSetReadMode.

Definition at line 2947 of file optixpp_namespace.h.

#### 2.17.2.21 void optix::TextureSamplerObj::setWrapMode ( unsigned int *dim*, RTwrapmode *wrapmode* ) [inline]

Set the texture wrap mode for this sampler. See rtTextureSamplerSetWrapMode.

Definition at line 2906 of file optixpp_namespace.h.

#### 2.17.2.22 void optix::TextureSamplerObj::unregisterGLTexture ( ) [inline]

Unregister the texture's buffer, re-enabling OptiX operations. See rtTextureSamplerGL-Unregister.

Definition at line 2993 of file optixpp_namespace.h.

#### 2.17.2.23 void optix::TextureSamplerObj::validate ( ) [inline, virtual]

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2870 of file optixpp_namespace.h.

### 2.17.3 Friends And Related Function Documentation

#### 2.17.3.1 friend class Handle< TextureSamplerObj > [friend]

Definition at line 1405 of file optixpp_namespace.h.

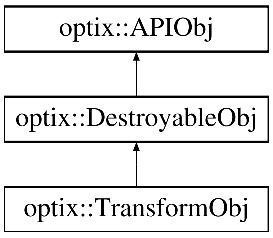
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.18 optix::TransformObj Class Reference

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::TransformObj:



### Public Member Functions

- void destroy ()
- void validate ()
- Context getContext () const
- RTtransform get ()

### Friends

- class Handle< TransformObj >

- template<typename T >
  void setChild (T child)
- template<typename T >
  T getChild () const

- void setMatrix (bool transpose, const float ∗matrix, const float ∗inverse_matrix)
- void getMatrix (bool transpose, float ∗matrix, float ∗inverse_matrix) const

### 2.18.1 Detailed Description

Transform wraps the OptiX C API RTtransform opaque type and its associated function set.

Definition at line 1012 of file optixpp_namespace.h.

### 2.18.2 Member Function Documentation

#### 2.18.2.1 void **optix::TransformObj::destroy** ( ) `[inline, virtual]`

call rt[ObjectType]Destroy on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2438 of file optixpp_namespace.h.

#### 2.18.2.2 RTtransform **optix::TransformObj::get** ( ) `[inline]`

Get the underlying OptiX C API RTtransform opaque pointer.

Definition at line 2481 of file optixpp_namespace.h.

#### 2.18.2.3 template<typename T > T **optix::TransformObj::getChild** ( ) const `[inline]`

Set the child node of this transform. See rtTransformGetChild.

Definition at line 2464 of file optixpp_namespace.h.

#### 2.18.2.4 **Context optix::TransformObj::getContext** ( ) const `[inline, virtual]`

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 2450 of file optixpp_namespace.h.

#### 2.18.2.5 void **optix::TransformObj::getMatrix** ( bool *transpose*, float ∗ *matrix*, float ∗ *inverse_matrix* ) const `[inline]`

Get the transform matrix for this node. See rtTransformGetMatrix.

Definition at line 2476 of file optixpp_namespace.h.

#### 2.18.2.6 template<typename T > void **optix::TransformObj::setChild** ( T *child* ) `[inline]`

Set the child node of this transform. See rtTransformSetChild.

Definition at line 2458 of file optixpp_namespace.h.

#### 2.18.2.7 void **optix::TransformObj::setMatrix** ( bool *transpose*, const float ∗ *matrix*, const float ∗ *inverse_matrix* ) `[inline]`

Set the transform matrix for this node. See rtTransformSetMatrix.

Definition at line 2471 of file optixpp_namespace.h.

#### 2.18.2.8 void **optix::TransformObj::validate** ( ) `[inline, virtual]`

call rt[ObjectType]Validate on the underlying OptiX C object

Implements optix::DestroyableObj.

Definition at line 2445 of file optixpp_namespace.h.

### 2.18.3 Friends And Related Function Documentation

#### 2.18.3.1 friend class **Handle**< **TransformObj** > `[friend]`

Definition at line 1040 of file optixpp_namespace.h.

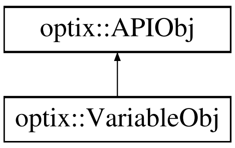
The documentation for this class was generated from the following file:

- optixpp_namespace.h

## 2.19 optix::VariableObj Class Reference

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::VariableObj:



#### Public Member Functions

- Context getContext () const
- std::string getName () const
- std::string getAnnotation () const
- RTobjecttype getType () const
- RTvariable get ()
- RTsize getSize () const

### Friends

- class Handle< VariableObj >

### Float setters

Set variable to have a float value.

- void setFloat (float f1)
- void setFloat (optix::float2 f)
- void setFloat (float f1, float f2)
- void setFloat (optix::float3 f)
- void setFloat (float f1, float f2, float f3)
- void setFloat (optix::float4 f)
- void setFloat (float f1, float f2, float f3, float f4)
- void set1fv (const float ∗f)
- void set2fv (const float ∗f)
- void set3fv (const float ∗f)
- void set4fv (const float ∗f)

### Int setters

Set variable to have an int value.

- void setInt (int i1)
- void setInt (int i1, int i2)
- void setInt (optix::int2 i)
- void setInt (int i1, int i2, int i3)
- void setInt (optix::int3 i)
- void setInt (int i1, int i2, int i3, int i4)
- void setInt (optix::int4 i)
- void set1iv (const int ∗i)
- void set2iv (const int ∗i)
- void set3iv (const int ∗i)
- void set4iv (const int ∗i)

### Unsigned int setters

Set variable to have an unsigned int value.

- void setUint (unsigned int u1)
- void setUint (unsigned int u1, unsigned int u2)
- void setUint (unsigned int u1, unsigned int u2, unsigned int u3)
- void setUint (unsigned int u1, unsigned int u2, unsigned int u3, unsigned int u4)
- void setUint (optix::uint2 u)

- void setUint (optix::uint3 u)
- void setUint (optix::uint4 u)
- void set1uiv (const unsigned int ∗u)
- void set2uiv (const unsigned int ∗u)
- void set3uiv (const unsigned int ∗u)
- void set4uiv (const unsigned int ∗u)

#### Matrix setters

Set variable to have a Matrix value

- void setMatrix2x2fv (bool transpose, const float ∗m)
- void setMatrix2x3fv (bool transpose, const float ∗m)
- void setMatrix2x4fv (bool transpose, const float ∗m)
- void setMatrix3x2fv (bool transpose, const float ∗m)
- void setMatrix3x3fv (bool transpose, const float ∗m)
- void setMatrix3x4fv (bool transpose, const float ∗m)
- void setMatrix4x2fv (bool transpose, const float ∗m)
- void setMatrix4x3fv (bool transpose, const float ∗m)
- void setMatrix4x4fv (bool transpose, const float ∗m)

#### Numeric value getters

Query value of a variable with scalar numeric value

- float getFloat () const
- unsigned int getUint () const
- int getInt () const

#### OptiX API object setters

Set variable to have an OptiX API object as its value

- void setBuffer (Buffer buffer)
- void set (Buffer buffer)
- void setTextureSampler (TextureSampler texturesample)
- void set (TextureSampler texturesample)
- void set (GeometryGroup group)
- void set (Group group)
- void set (Program program)
- void set (Selector selector)
- void set (Transform transform)

#### OptiX API object getters

Reitrieve OptiX API object value from a variable

- Buffer getBuffer () const
- TextureSampler getTextureSampler () const
- Program getProgram () const

#### User data variable accessors

- void setUserData (RTsize size, const void ∗ptr)
- void getUserData (RTsize size, void ∗ptr) const

### 2.19.1 Detailed Description

Variable object wraps OptiX C API RTvariable type and its related function set.

See OptiX programming guide and API reference for complete description of the usage and behavior of RTvariable objects. Creation and querying of Variables can be performed via the Handle::operator[] function of the scope object associated with the variable. For example:

```
my_context["new_variable"]->setFloat( 1.0f );
```

will create a variable named `new_variable` on the object `my_context` if it does not already exist. It will then set the value of that variable to be a float 1.0f.

Definition at line 406 of file optixpp_namespace.h.

### 2.19.2 Member Function Documentation

#### 2.19.2.1 RTvariable optix::VariableObj::get ( ) `[inline]`

Get the OptiX C API object wrapped by this instance.

Definition at line 3550 of file optixpp_namespace.h.

#### 2.19.2.2 std::string optix::VariableObj::getAnnotation ( ) const `[inline]`

Retrieve the annotation associated with the variable.

Definition at line 3536 of file optixpp_namespace.h.

#### 2.19.2.3 Buffer optix::VariableObj::getBuffer ( ) const `[inline]`

Definition at line 3521 of file optixpp_namespace.h.

#### 2.19.2.4 Context optix::VariableObj::getContext ( ) const [inline, virtual]

Retrieve the context this object is associated with. See rt[ObjectType]GetContext.

Implements optix::APIObj.

Definition at line 3227 of file optixpp_namespace.h.

#### 2.19.2.5 float optix::VariableObj::getFloat ( ) const [inline]

Definition at line 3445 of file optixpp_namespace.h.

#### 2.19.2.6 int optix::VariableObj::getInt ( ) const [inline]

Definition at line 3459 of file optixpp_namespace.h.

#### 2.19.2.7 std::string optix::VariableObj::getName ( ) const [inline]

Retrieve the name of the variable.

Definition at line 3529 of file optixpp_namespace.h.

#### 2.19.2.8 optix::Program optix::VariableObj::getProgram ( ) const [inline]

Definition at line 3570 of file optixpp_namespace.h.

#### 2.19.2.9 RTsize optix::VariableObj::getSize ( ) const [inline]

Get the size of the variable data in bytes (eg, float4 returns 4∗sizeof(float) )

Definition at line 3555 of file optixpp_namespace.h.

#### 2.19.2.10 optix::TextureSampler optix::VariableObj::getTextureSampler ( ) const [inline]

Definition at line 3562 of file optixpp_namespace.h.

#### 2.19.2.11 RTobjecttype optix::VariableObj::getType ( ) const [inline]

Query the object type of the variable.

Definition at line 3543 of file optixpp_namespace.h.

#### 2.19.2.12 unsigned int optix::VariableObj::getUint ( ) const [inline]

Definition at line 3452 of file optixpp_namespace.h.

#### 2.19.2.13 void optix::VariableObj::getUserData ( RTsize *size*, void ∗ *ptr* ) const [inline]

Retrieve a user defined type given the sizeof the user object.

Definition at line 3481 of file optixpp_namespace.h.

#### 2.19.2.14 void optix::VariableObj::set ( Buffer *buffer* ) [inline]

Definition at line 3471 of file optixpp_namespace.h.

#### 2.19.2.15 void optix::VariableObj::set ( TextureSampler *texturesample* )

#### 2.19.2.16 void optix::VariableObj::set ( GeometryGroup *group* )

#### 2.19.2.17 void optix::VariableObj::set ( Group *group* )

#### 2.19.2.18 void optix::VariableObj::set ( Program *program* )

#### 2.19.2.19 void optix::VariableObj::set ( Selector *selector* )

#### 2.19.2.20 void optix::VariableObj::set ( Transform *transform* )

#### 2.19.2.21 void optix::VariableObj::set1fv ( const float ∗ *f* ) [inline]

Set variable value to a scalar float.

Definition at line 3369 of file optixpp_namespace.h.

#### 2.19.2.22 void optix::VariableObj::set1iv ( const int ∗ *i* ) [inline]

Definition at line 3425 of file optixpp_namespace.h.

#### 2.19.2.23 void optix::VariableObj::set1uiv ( const unsigned int ∗ *u* ) [inline]

Definition at line 3269 of file optixpp_namespace.h.

#### 2.19.2.24 void optix::VariableObj::set2fv ( const float ∗ *f* ) [inline]

Set variable value to a float2.

Definition at line 3374 of file optixpp_namespace.h.

#### 2.19.2.25 void optix::VariableObj::set2iv ( const int ∗ *i* ) [inline]

Definition at line 3430 of file optixpp_namespace.h.

#### 2.19.2.26 void optix::VariableObj::set2uiv ( const unsigned int ∗ *u* ) [inline]

Definition at line 3274 of file optixpp_namespace.h.

#### 2.19.2.27 void optix::VariableObj::set3fv ( const float ∗ *f* ) [inline]

Set variable value to a float3.

Definition at line 3379 of file optixpp_namespace.h.

#### 2.19.2.28 void optix::VariableObj::set3iv ( const int ∗ *i* ) [inline]

Definition at line 3435 of file optixpp_namespace.h.

#### 2.19.2.29 void optix::VariableObj::set3uiv ( const unsigned int ∗ *u* ) [inline]

Definition at line 3279 of file optixpp_namespace.h.

#### 2.19.2.30 void optix::VariableObj::set4fv ( const float ∗ *f* ) [inline]

Set variable value to a float4.

Definition at line 3384 of file optixpp_namespace.h.

#### 2.19.2.31 void optix::VariableObj::set4iv ( const int ∗ *i* ) [inline]

Definition at line 3440 of file optixpp_namespace.h.

#### 2.19.2.32 void optix::VariableObj::set4uiv ( const unsigned int ∗ *u* ) [inline]

Definition at line 3284 of file optixpp_namespace.h.

#### 2.19.2.33 void optix::VariableObj::setBuffer ( Buffer *buffer* ) [inline]

Definition at line 3466 of file optixpp_namespace.h.

#### 2.19.2.34 void optix::VariableObj::setFloat ( float *f1* ) [inline]

Set variable value to a scalar float.

Definition at line 3334 of file optixpp_namespace.h.

#### 2.19.2.35 void optix::VariableObj::setFloat ( optix::float2 *f* ) [inline]

Set variable value to a float2.

Definition at line 3339 of file optixpp_namespace.h.

#### 2.19.2.36 void optix::VariableObj::setFloat ( float *f1*, float *f2* ) [inline]

Set variable value to a float2.

Definition at line 3344 of file optixpp_namespace.h.

#### 2.19.2.37 void optix::VariableObj::setFloat ( optix::float3 *f* ) [inline]

Set variable value to a float3.

Definition at line 3349 of file optixpp_namespace.h.

#### 2.19.2.38 void optix::VariableObj::setFloat ( float *f1*, float *f2*, float *f3* ) [inline]

Set variable value to a float3.

Definition at line 3354 of file optixpp_namespace.h.

#### 2.19.2.39 void optix::VariableObj::setFloat ( optix::float4 *f* ) [inline]

Set variable value to a float4.

Definition at line 3359 of file optixpp_namespace.h.

#### 2.19.2.40 void optix::VariableObj::setFloat ( float *f1*, float *f2*, float *f3*, float *f4* ) [inline]

Set variable value to a float4.

Definition at line 3364 of file optixpp_namespace.h.

#### 2.19.2.41 void optix::VariableObj::setInt ( int *i1* ) [inline]

Definition at line 3390 of file optixpp_namespace.h.

#### 2.19.2.42 void optix::VariableObj::setInt ( int *i1*, int *i2* ) [inline]

Definition at line 3400 of file optixpp_namespace.h.

#### 2.19.2.43 void optix::VariableObj::setInt ( optix::int2 *i* ) [inline]

Definition at line 3395 of file optixpp_namespace.h.

#### 2.19.2.44 void optix::VariableObj::setInt ( int *i1*, int *i2*, int *i3* ) [inline]

Definition at line 3410 of file optixpp_namespace.h.

#### 2.19.2.45 void optix::VariableObj::setInt ( optix::int3 *i* ) [inline]

Definition at line 3405 of file optixpp_namespace.h.

#### 2.19.2.46 void optix::VariableObj::setInt ( int *i1*, int *i2*, int *i3*, int *i4* ) [inline]

Definition at line 3420 of file optixpp_namespace.h.

#### 2.19.2.47 void optix::VariableObj::setInt ( optix::int4 *i* ) [inline]

Definition at line 3415 of file optixpp_namespace.h.

#### 2.19.2.48 void optix::VariableObj::setMatrix2x2fv ( bool *transpose*, const float ∗ *m* ) [inline]

Definition at line 3289 of file optixpp_namespace.h.

#### 2.19.2.49 void optix::VariableObj::setMatrix2x3fv ( bool *transpose*, const float ∗ *m* ) [inline]

Definition at line 3294 of file optixpp_namespace.h.

#### 2.19.2.50 void optix::VariableObj::setMatrix2x4fv ( bool *transpose*, const float ∗ *m* ) [inline]

Definition at line 3299 of file optixpp_namespace.h.

#### 2.19.2.51 void **optix::VariableObj::setMatrix3x2fv** ( bool *transpose*, const float ∗ *m* ) `[inline]`

Definition at line 3304 of file optixpp_namespace.h.

#### 2.19.2.52 void **optix::VariableObj::setMatrix3x3fv** ( bool *transpose*, const float ∗ *m* ) `[inline]`

Definition at line 3309 of file optixpp_namespace.h.

#### 2.19.2.53 void **optix::VariableObj::setMatrix3x4fv** ( bool *transpose*, const float ∗ *m* ) `[inline]`

Definition at line 3314 of file optixpp_namespace.h.

#### 2.19.2.54 void **optix::VariableObj::setMatrix4x2fv** ( bool *transpose*, const float ∗ *m* ) `[inline]`

Definition at line 3319 of file optixpp_namespace.h.

#### 2.19.2.55 void **optix::VariableObj::setMatrix4x3fv** ( bool *transpose*, const float ∗ *m* ) `[inline]`

Definition at line 3324 of file optixpp_namespace.h.

#### 2.19.2.56 void **optix::VariableObj::setMatrix4x4fv** ( bool *transpose*, const float ∗ *m* ) `[inline]`

Definition at line 3329 of file optixpp_namespace.h.

#### 2.19.2.57 void **optix::VariableObj::setTextureSampler** ( **TextureSampler** *texturesample* ) `[inline]`

Definition at line 3486 of file optixpp_namespace.h.

#### 2.19.2.58 void **optix::VariableObj::setUint** ( unsigned int *u1* ) `[inline]`

Definition at line 3234 of file optixpp_namespace.h.

#### 2.19.2.59 void **optix::VariableObj::setUint** ( unsigned int *u1*, unsigned int *u2* ) `[inline]`

Definition at line 3239 of file optixpp_namespace.h.

#### 2.19.2.60 void **optix::VariableObj::setUint** ( unsigned int *u1*, unsigned int *u2*, unsigned int *u3* ) `[inline]`

Definition at line 3244 of file optixpp_namespace.h.

#### 2.19.2.61 void optix::VariableObj::setUint ( unsigned int *u1*, unsigned int *u2*, unsigned int *u3*, unsigned int *u4* ) [inline]

Definition at line 3249 of file optixpp_namespace.h.

#### 2.19.2.62 void optix::VariableObj::setUint ( optix::uint2 *u* ) [inline]

Definition at line 3254 of file optixpp_namespace.h.

#### 2.19.2.63 void optix::VariableObj::setUint ( optix::uint3 *u* ) [inline]

Definition at line 3259 of file optixpp_namespace.h.

#### 2.19.2.64 void optix::VariableObj::setUint ( optix::uint4 *u* ) [inline]

Definition at line 3264 of file optixpp_namespace.h.

#### 2.19.2.65 void optix::VariableObj::setUserData ( RTsize *size*, const void ∗ *ptr* ) [inline]

Set the variable to a user defined type given the sizeof the user object.

Definition at line 3476 of file optixpp_namespace.h.

### 2.19.3 Friends And Related Function Documentation

#### 2.19.3.1 friend class Handle< VariableObj > [friend]

Definition at line 593 of file optixpp_namespace.h.

The documentation for this class was generated from the following file:

- optixpp_namespace.h

# 3 File Documentation

## 3.1 optixpp_namespace.h File Reference

```
#include "../optix.h"  #include "../optix_gl_interop.h"×
#include "../optix_cuda_interop.h"  #include <string>×
#include <vector>  #include <iterator>  #include "optixu-
_vector_types.h"
```

### Classes

- class optix::Handle< T >

  *The Handle class is a reference counted handle class used to manipulate API objects.*
- class optix::Exception

*Exception class for error reporting from the OptiXpp API.*

- class optix::APIObj

  *Base class for all reference counted wrappers around OptiX C API opaque types.*
- class optix::DestroyableObj

  *Base class for all wrapper objects which can be destroyed and validated.*
- class optix::ScopedObj

  *Base class for all objects which are OptiX variable containers.*
- class optix::VariableObj

  *Variable object wraps OptiX C API RTvariable type and its related function set.*
- class optix::ContextObj

  *Context object wraps the OptiX C API RTcontext opaque type and its associated function set.*
- class optix::ProgramObj

  *Program object wraps the OptiX C API RTprogram opaque type and its associated function set.*
- class optix::GroupObj

  *Group wraps the OptiX C API RTgroup opaque type and its associated function set.*
- class optix::GeometryGroupObj

  *GeometryGroup wraps the OptiX C API RTgeometrygroup opaque type and its associated function set.*
- class optix::TransformObj

  *Transform wraps the OptiX C API RTtransform opaque type and its associated function set.*
- class optix::SelectorObj

  *Selector wraps the OptiX C API RTselector opaque type and its associated function set.*
- class optix::AccelerationObj

  *Acceleration wraps the OptiX C API RTacceleration opaque type and its associated function set.*
- class optix::GeometryInstanceObj

  *GeometryInstance wraps the OptiX C API RTgeometryinstance acceleration opaque type and its associated function set.*
- class optix::GeometryObj

  *Geometry wraps the OptiX C API RTgeometry opaque type and its associated function set.*
- class optix::MaterialObj

  *Material wraps the OptiX C API RTmaterial opaque type and its associated function set.*
- class optix::TextureSamplerObj

  *TextureSampler wraps the OptiX C API RTtexturesampler opaque type and its associated function set.*
- class optix::BufferObj

  *Buffer wraps the OptiX C API RTbuffer opaque type and its associated function set.*

### Typedefs

- typedef Handle< AccelerationObj > optix::Acceleration
- typedef Handle< BufferObj > optix::Buffer
- typedef Handle< ContextObj > optix::Context
- typedef Handle< GeometryObj > optix::Geometry
- typedef Handle< GeometryGroupObj > optix::GeometryGroup
- typedef Handle < GeometryInstanceObj > optix::GeometryInstance
- typedef Handle< GroupObj > optix::Group
- typedef Handle< MaterialObj > optix::Material
- typedef Handle< ProgramObj > optix::Program
- typedef Handle< SelectorObj > optix::Selector
- typedef Handle< TextureSamplerObj > optix::TextureSampler
- typedef Handle< TransformObj > optix::Transform
- typedef Handle< VariableObj > optix::Variable

### 3.1.1 Detailed Description

A C++ wrapper around the OptiX API.

Definition in file optixpp_namespace.h.

## 3.2 optixpp_namespace.h

```

/*
 * Copyright (c) 2008 - 2009 NVIDIA Corporation.  All rights reserved.
 *
 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
 * rights in and to this software, related documentation and any modifications thereto.
 * Any use, reproduction, disclosure or distribution of this software and related
 * documentation without an express license agreement from NVIDIA Corporation is strictly
 * prohibited.
 *
 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGES
 */




```

```

#ifndef __optixu_optixpp_namespace_h__
#define __optixu_optixpp_namespace_h__

#include "../optix.h"

#ifdef _WIN32
#  ifndef WIN32_LEAN_AND_MEAN
#    define WIN32_LEAN_AND_MEAN
#  endif
#  include <windows.h>
#  include "../optix_d3d9_interop.h"
#  include "../optix_d3d10_interop.h"
#  include "../optix_d3d11_interop.h"
#endif
#include "../optix_gl_interop.h"
#include "../optix_cuda_interop.h"

#include <string>
#include <vector>
#include <iterator>
#include "optixu_vector_types.h"

//-----------------------------------------------------------------------------
//
// Doxygen group specifications
//
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
//
// C++ API
//
//-----------------------------------------------------------------------------

namespace optix {

  class AccelerationObj;
  class BufferObj;
  class ContextObj;
  class GeometryObj;
  class GeometryGroupObj;
  class GeometryInstanceObj;
  class GroupObj;
  class MaterialObj;
  class ProgramObj;
  class SelectorObj;
  class TextureSamplerObj;
  class TransformObj;
  class VariableObj;

  class APIObj;
  class ScopedObj;


  template<class T>
  class Handle {
  public:
    Handle() : ptr(0) {}

    Handle(T* ptr) : ptr(ptr) { ref(); }

    template<class U>
    Handle(U* ptr) : ptr(ptr) { ref(); }

    Handle(const Handle<T>& copy) : ptr(copy.ptr) { ref(); }

    template<class U>
    Handle(const Handle<U>& copy) : ptr(copy.ptr) { ref(); }

    Handle<T>& operator=(const Handle<T>& copy)
    { if(ptr != copy.ptr) { unref(); ptr = copy.ptr; ref(); } return *this; }

    template<class U>
    Handle<T>& operator=( const Handle<U>& copy)
    { if(ptr != copy.ptr) { unref(); ptr = copy.ptr; ref(); } return *this; }

    ~Handle() { unref(); }

    static Handle<T> take( typename T::api_t p ) { return p? new T(p) : 0; }
    static Handle<T> take( RTobject p ) { return p? new T(static_cast<typename
 T::api_t>(p)) : 0; }

          T* operator->()           { return ptr; }
    const T* operator->() const     { return ptr; }

          T* get()                  { return ptr; }
    const T* get() const            { return ptr; }

    operator bool() const  { return ptr != 0; }

    Handle<VariableObj> operator[](const std::string& varname);

    Handle<VariableObj> operator[](const char* varname);

    static Handle<T> create() { return T::create(); }

    static unsigned int getDeviceCount() { return T::getDeviceCount(); }

  private:
    inline void ref() { if(ptr) ptr->addReference(); }
    inline void unref() { if(ptr && ptr->removeReference() == 0) delete ptr; }
    T* ptr;
  };


  //
-----------------------------------------------------------------------------

  typedef Handle<AccelerationObj>     Acceleration;
  typedef Handle<BufferObj>           Buffer;
  typedef Handle<ContextObj>          Context;
  typedef Handle<GeometryObj>         Geometry;
  typedef Handle<GeometryGroupObj>    GeometryGroup;
  typedef Handle<GeometryInstanceObj> GeometryInstance;
  typedef Handle<GroupObj>            Group;
  typedef Handle<MaterialObj>         Material;
  typedef Handle<ProgramObj>          Program;
  typedef Handle<SelectorObj>         Selector;
  typedef Handle<TextureSamplerObj>   TextureSampler;
  typedef Handle<TransformObj>        Transform;
  typedef Handle<VariableObj>         Variable;


  //
-----------------------------------------------------------------------------


  class Exception: public std::exception {
  public:
    Exception( const std::string& message, RTresult error_code =
RT_ERROR_UNKNOWN )
      : m_message(message), m_error_code( error_code ) {}

    virtual ~Exception() throw() {}

    const std::string& getErrorString() const { return m_message; }

    RTresult getErrorCode() const { return m_error_code; }

    static Exception makeException( RTresult code, RTcontext context );

    virtual const char* what() const throw() { return getErrorString().c_str();
 }
  private:
    std::string m_message;
    RTresult    m_error_code;
  };
```

```

inline Exception Exception::makeException( RTresult code, RTcontext context )
{
  const char* str;
  rtContextGetErrorString( context, code, &str);
  return Exception( std::string(str), code );
}


//
-----------------------------------------------------------------------------


class APIObj {
public:
  APIObj() : ref_count(0) {}
  virtual ~APIObj() {}

  void addReference()    { ++ref_count; }
  int  removeReference() { return --ref_count; }

  virtual Context getContext()const=0;

  virtual void checkError(RTresult code)const;
  virtual void checkError(RTresult code, Context context )const;

  void checkErrorNoGetContext(RTresult code)const;

  static Exception makeException( RTresult code, RTcontext context );
private:
  int ref_count;
};

inline Exception APIObj::makeException( RTresult code, RTcontext context )
{
  return Exception::makeException( code, context );
}


//
-----------------------------------------------------------------------------


class DestroyableObj : public APIObj {
public:
  virtual ~DestroyableObj() {}

  virtual void destroy() = 0;

  virtual void validate() = 0;
};



//
-----------------------------------------------------------------------------


class ScopedObj : public DestroyableObj {
public:
  virtual ~ScopedObj() {}

  virtual Variable declareVariable(const std::string& name) = 0;
  virtual Variable queryVariable(const std::string& name) const = 0;
  virtual void removeVariable(Variable v) = 0;
  virtual unsigned int getVariableCount() const = 0;
  virtual Variable getVariable(unsigned int index) const = 0;
};



//
-----------------------------------------------------------------------------

```

```
  class VariableObj : public APIObj {
  public:

    Context getContext() const;



    void setFloat(float f1);
    void setFloat(optix::float2 f);
    void setFloat(float f1, float f2);
    void setFloat(optix::float3 f);
    void setFloat(float f1, float f2, float f3);
    void setFloat(optix::float4 f);
    void setFloat(float f1, float f2, float f3, float f4);
    void set1fv(const float* f);
    void set2fv(const float* f);
    void set3fv(const float* f);
    void set4fv(const float* f);


    void setInt(int i1);
    void setInt(int i1, int i2);
    void setInt(optix::int2 i);
    void setInt(int i1, int i2, int i3);
    void setInt(optix::int3 i);
    void setInt(int i1, int i2, int i3, int i4);
    void setInt(optix::int4 i);
    void set1iv(const int* i);
    void set2iv(const int* i);
    void set3iv(const int* i);
    void set4iv(const int* i);


    void setUint(unsigned int u1);
    void setUint(unsigned int u1, unsigned int u2);
    void setUint(unsigned int u1, unsigned int u2, unsigned int u3);
    void setUint(unsigned int u1, unsigned int u2, unsigned int u3, unsigned
 int u4);
    void setUint(optix::uint2 u);
    void setUint(optix::uint3 u);
    void setUint(optix::uint4 u);
    void set1uiv(const unsigned int* u);
    void set2uiv(const unsigned int* u);
    void set3uiv(const unsigned int* u);
    void set4uiv(const unsigned int* u);


    void setMatrix2x2fv(bool transpose, const float* m);
    void setMatrix2x3fv(bool transpose, const float* m);
    void setMatrix2x4fv(bool transpose, const float* m);
    void setMatrix3x2fv(bool transpose, const float* m);
    void setMatrix3x3fv(bool transpose, const float* m);
    void setMatrix3x4fv(bool transpose, const float* m);
    void setMatrix4x2fv(bool transpose, const float* m);
    void setMatrix4x3fv(bool transpose, const float* m);
    void setMatrix4x4fv(bool transpose, const float* m);


    float getFloat() const;
    unsigned int getUint() const;
    int getInt() const;

#if 0
    // Not implemented yet...

    // The getFloat functions can be overloaded by parameter type.
    void getFloat(float* f);
    void getFloat(float* f1, float* f2);
    void getFloat(optix::float2* f);
    void getFloat(float* f1, float* f2, float* f3);
    void getFloat(optix::float3* f);
    void getFloat(float* f1, float* f2, float* f3, float* f4);
    void getFloat(optix::float4* f);
    // This one will need a different name to distinguish it from 'float getFloat()'.
    optix::float2 getFloat2();
    optix::float3 getFloat3();
    optix::float4 getFloat4();

    void get1fv(float* f);
    void get2fv(float* f);
    void get3fv(float* f);
    void get4fv(float* f);

    get1i (int* i1);
    get2i (int* i1, int* i2);
    get3i (int* i1, int* i2, int* i3);
    get4i (int* i1, int* i2, int* i3, int* i4);
    get1iv(int* i);
    get2iv(int* i);
    get3iv(int* i);
    get4iv(int* i);

    get1ui (unsigned int* u1);
    get2ui (unsigned int* u1, unsigned int* u2);
    get3ui (unsigned int* u1, unsigned int* u2, unsigned int* u3);
    get4ui (unsigned int* u1, unsigned int* u2, unsigned int* u3, unsigned int* u4);
    get1uiv(unsigned int* u);
    get2uiv(unsigned int* u);
    get3uiv(unsigned int* u);
    get4uiv(unsigned int* u);

    getMatrix2x2fv(bool transpose, float* m);
    getMatrix2x3fv(bool transpose, float* m);
    getMatrix2x4fv(bool transpose, float* m);
    getMatrix3x2fv(bool transpose, float* m);
    getMatrix3x3fv(bool transpose, float* m);
    getMatrix3x4fv(bool transpose, float* m);
    getMatrix4x2fv(bool transpose, float* m);
    getMatrix4x3fv(bool transpose, float* m);
    getMatrix4x4fv(bool transpose, float* m);
#endif



    void setBuffer(Buffer buffer);
    void set(Buffer buffer);
    void setTextureSampler(TextureSampler texturesample);
    void set(TextureSampler texturesample);
    void set(GeometryGroup group);
    void set(Group group);
    void set(Program program);
    void set(Selector selector);
    void set(Transform transform);


    Buffer getBuffer() const;
    TextureSampler getTextureSampler() const;
    Program getProgram() const;



    void setUserData(RTsize size, const void* ptr);
    void getUserData(RTsize size,       void* ptr) const;

    std::string getName() const;

    std::string getAnnotation() const;

    RTobjecttype getType() const;

    RTvariable get();

    RTsize getSize() const;

  private:
```

```
  typedef RTvariable api_t;

  RTvariable m_variable;
  VariableObj(RTvariable variable) : m_variable(variable) {}
  friend class Handle<VariableObj>;

};

template<class T>
Handle<VariableObj> Handle<T>::operator[](const std::string& varname)
{
  Variable v = ptr->queryVariable( varname );
  if( v.operator->() == 0)
    v = ptr->declareVariable( varname );
  return v;
}

template<class T>
Handle<VariableObj> Handle<T>::operator[](const char* varname)
{
  return (*this)[ std::string( varname ) ];
}


//----------------------------------------------------------------------------


class ContextObj : public ScopedObj {
public:

  static unsigned int getDeviceCount();

  static std::string getDeviceName(int ordinal);

  static void getDeviceAttribute(int ordinal, RTdeviceattribute attrib, RTsize size, void* p);

  static Context create();

  void destroy();

  void validate();

  Context getContext() const;

  void checkError(RTresult code)const;

  std::string getErrorString( RTresult code ) const;

  Acceleration createAcceleration(const char* builder, const char* traverser);

  Buffer createBuffer(unsigned int type);
  Buffer createBuffer(unsigned int type, RTformat format);
  Buffer createBuffer(unsigned int type, RTformat format, RTsize width);
  Buffer createBuffer(unsigned int type, RTformat format, RTsize width, RTsize height);
  Buffer createBuffer(unsigned int type, RTformat format, RTsize width, RTsize height, RTsize depth);

  Buffer createBufferForCUDA(unsigned int type);
  Buffer createBufferForCUDA(unsigned int type, RTformat format);
  Buffer createBufferForCUDA(unsigned int type, RTformat format, RTsize width);
  Buffer createBufferForCUDA(unsigned int type, RTformat format, RTsize width, RTsize height);
  Buffer createBufferForCUDA(unsigned int type, RTformat format, RTsize width, RTsize height, RTsize depth);

  Buffer createBufferFromGLBO(unsigned int type, unsigned int vbo);

  TextureSampler createTextureSamplerFromGLImage(unsigned int id, RTgltarget target);
```

```

#ifdef _WIN32

    Buffer createBufferFromD3D9Resource(unsigned int type, IDirect3DResource9 * pResource);
    Buffer createBufferFromD3D10Resource(unsigned int type, ID3D10Resource * pResource);
    Buffer createBufferFromD3D11Resource(unsigned int type, ID3D11Resource * pResource);

    TextureSampler createTextureSamplerFromD3D9Resource(IDirect3DResource9 * pResource);
    TextureSampler createTextureSamplerFromD3D10Resource(ID3D10Resource * pResource);
    TextureSampler createTextureSamplerFromD3D11Resource(ID3D11Resource * pResource);
#endif

    Geometry createGeometry();
    GeometryInstance createGeometryInstance();
    template<class Iterator>
    GeometryInstance createGeometryInstance( Geometry geometry, Iterator matlbegin, Iterator matlend );

    Group createGroup();
    template<class Iterator>
    Group createGroup( Iterator childbegin, Iterator childend );

    GeometryGroup createGeometryGroup();
    template<class Iterator>
    GeometryGroup createGeometryGroup( Iterator childbegin, Iterator childend )
;

    Transform createTransform();

    Material createMaterial();

    Program createProgramFromPTXFile  ( const std::string& ptx, const std::string& program_name );
    Program createProgramFromPTXString( const std::string& ptx, const std::string& program_name );

    Selector createSelector();

    TextureSampler createTextureSampler();


    template<class Iterator>
    void setDevices(Iterator begin, Iterator end);

#ifdef _WIN32

    void setD3D9Device(IDirect3DDevice9* device);
    void setD3D10Device(ID3D10Device* device);
    void setD3D11Device(ID3D11Device* device);
#endif

    std::vector<int> getEnabledDevices() const;

    unsigned int getEnabledDeviceCount() const;

    int getMaxTextureCount() const;

    int getCPUNumThreads() const;

    RTsize getUsedHostMemory() const;

    int getGPUPagingActive() const;

    int getGPUPagingForcedOff() const;

    RTsize getAvailableDeviceMemory(int ordinal) const;

    void setCPUNumThreads(int cpu_num_threads);

    void setGPUPagingForcedOff(int gpu_paging_forced_off);

    void setStackSize(RTsize  stack_size_bytes);
    RTsize getStackSize() const;

    void setTimeoutCallback(RTtimeoutcallback callback, double
  min_polling_seconds);

    void setEntryPointCount(unsigned int  num_entry_points);
    unsigned int getEntryPointCount() const;

    void setRayTypeCount(unsigned int  num_ray_types);
    unsigned int getRayTypeCount() const;

    void setRayGenerationProgram(unsigned int entry_point_index, Program
  program);
    Program getRayGenerationProgram(unsigned int entry_point_index) const;

    void setExceptionProgram(unsigned int entry_point_index, Program  program);
    Program getExceptionProgram(unsigned int entry_point_index) const;

    void setExceptionEnabled( RTexception exception, bool enabled );
    bool getExceptionEnabled( RTexception exception ) const;

    void setMissProgram(unsigned int ray_type_index, Program  program);
    Program getMissProgram(unsigned int ray_type_index) const;

    void compile();

    void launch(unsigned int entry_point_index, RTsize image_width);
    void launch(unsigned int entry_point_index, RTsize image_width, RTsize
  image_height);
    void launch(unsigned int entry_point_index, RTsize image_width, RTsize
  image_height, RTsize image_depth);

    int getRunningState() const;

    void setPrintEnabled(bool enabled);
    bool getPrintEnabled() const;
    void setPrintBufferSize(RTsize buffer_size_bytes);
    RTsize getPrintBufferSize() const;
    void setPrintLaunchIndex(int x, int y=-1, int z=-1);
    optix::int3 getPrintLaunchIndex() const;

    Variable declareVariable (const std::string& name);
    Variable queryVariable   (const std::string& name) const;
    void     removeVariable  (Variable v);
    unsigned int getVariableCount() const;
    Variable getVariable     (unsigned int index) const;

    RTcontext get();
  private:
    typedef RTcontext api_t;

    virtual ~ContextObj() {}
    RTcontext m_context;
    ContextObj(RTcontext context) : m_context(context) {}
    friend class Handle<ContextObj>;
  };



  //
  ---------------------------------------------------------------------------


  class ProgramObj : public ScopedObj {
  public:
    void destroy();
    void validate();

    Context getContext() const;

    Variable declareVariable (const std::string& name);
    Variable queryVariable   (const std::string& name) const;
    void     removeVariable  (Variable v);
    unsigned int getVariableCount() const;
    Variable getVariable     (unsigned int index) const;

    RTprogram get();
  private:
    typedef RTprogram api_t;
    virtual ~ProgramObj() {}
    RTprogram m_program;
    ProgramObj(RTprogram program) : m_program(program) {}
    friend class Handle<ProgramObj>;
  };


  //
------------------------------------------------------------------------------


  class GroupObj : public DestroyableObj {
  public:
    void destroy();
    void validate();

    Context getContext() const;

    void setAcceleration(Acceleration acceleration);
    Acceleration getAcceleration() const;

    void setChildCount(unsigned int  count);
    unsigned int getChildCount() const;

    template< typename T > void setChild(unsigned int index, T child);
    template< typename T > T getChild(unsigned int index) const;

    RTgroup get();

  private:
    typedef RTgroup api_t;
    virtual ~GroupObj() {}
    RTgroup m_group;
    GroupObj(RTgroup group) : m_group(group) {}
    friend class Handle<GroupObj>;
  };


  //
------------------------------------------------------------------------------


  class GeometryGroupObj : public DestroyableObj {
  public:
    void destroy();
    void validate();
    Context getContext() const;

    void setAcceleration(Acceleration acceleration);
    Acceleration getAcceleration() const;

    void setChildCount(unsigned int  count);
    unsigned int getChildCount() const;

    void setChild(unsigned int index, GeometryInstance geometryinstance);
    GeometryInstance getChild(unsigned int index) const;

    RTgeometrygroup get();

  private:
    typedef RTgeometrygroup api_t;
    virtual ~GeometryGroupObj() {}
    RTgeometrygroup m_geometrygroup;
    GeometryGroupObj(RTgeometrygroup geometrygroup) : m_geometrygroup(
geometrygroup) {}
    friend class Handle<GeometryGroupObj>;
  };


  //
-----------------------------------------------------------------------------


  class TransformObj  : public DestroyableObj {
  public:
    void destroy();
    void validate();
    Context getContext() const;

    template< typename T > void setChild(T child);
    template< typename T > T getChild() const;

    void setMatrix(bool transpose, const float* matrix, const float*
inverse_matrix);
    void getMatrix(bool transpose, float* matrix, float* inverse_matrix) const;

    RTtransform get();

  private:
    typedef RTtransform api_t;
    virtual ~TransformObj() {}
    RTtransform m_transform;
    TransformObj(RTtransform transform) : m_transform(transform) {}
    friend class Handle<TransformObj>;
  };


  //
-----------------------------------------------------------------------------


  class SelectorObj : public DestroyableObj {
  public:
    void destroy();
    void validate();
    Context getContext() const;

    void setVisitProgram(Program  program);
    Program getVisitProgram() const;

    void setChildCount(unsigned int  count);
    unsigned int getChildCount() const;

    template< typename T > void setChild(unsigned int index, T child);
    template< typename T > T getChild(unsigned int index) const;

    Variable declareVariable (const std::string& name);
    Variable queryVariable   (const std::string& name) const;
    void     removeVariable  (Variable v);
    unsigned int getVariableCount() const;
    Variable getVariable     (unsigned int index) const;

    RTselector get();

  private:
    typedef RTselector api_t;
    virtual ~SelectorObj() {}
    RTselector m_selector;
    SelectorObj(RTselector selector) : m_selector(selector) {}
    friend class Handle<SelectorObj>;
  };


  //
-----------------------------------------------------------------------------


  class AccelerationObj : public DestroyableObj {
  public:
    void destroy();
    void validate();
    Context getContext() const;

    void markDirty();
    bool isDirty() const;

    void        setProperty( const std::string& name, const std::string& value
);
    std::string getProperty( const std::string& name ) const;

    void        setBuilder(const std::string& builder);
    std::string getBuilder() const;
    void        setTraverser(const std::string& traverser);
    std::string getTraverser() const;

    RTsize getDataSize() const;
    void   getData( void* data ) const;
    void   setData( const void* data, RTsize size );

    RTacceleration get();

  private:
    typedef RTacceleration api_t;
    virtual ~AccelerationObj() {}
    RTacceleration m_acceleration;
    AccelerationObj(RTacceleration acceleration) : m_acceleration(acceleration)
 {}
    friend class Handle<AccelerationObj>;
  };


  //
---------------------------------------------------------------------------


  class GeometryInstanceObj : public ScopedObj {
  public:
    void destroy();
    void validate();
    Context getContext() const;

    void setGeometry(Geometry  geometry);
    Geometry getGeometry() const;

    void setMaterialCount(unsigned int  count);
    unsigned int getMaterialCount() const;

    void setMaterial(unsigned int idx, Material  material);
    Material getMaterial(unsigned int idx) const;

    unsigned int addMaterial(Material material);

    Variable declareVariable (const std::string& name);
    Variable queryVariable   (const std::string& name) const;
    void     removeVariable  (Variable v);
    unsigned int getVariableCount() const;
    Variable getVariable     (unsigned int index) const;

    RTgeometryinstance get();

  private:
    typedef RTgeometryinstance api_t;
    virtual ~GeometryInstanceObj() {}
    RTgeometryinstance m_geometryinstance;
    GeometryInstanceObj(RTgeometryinstance geometryinstance) :
m_geometryinstance(geometryinstance) {}
    friend class Handle<GeometryInstanceObj>;
  };


  //
---------------------------------------------------------------------------


  class GeometryObj : public ScopedObj {
```

```
  public:
    void destroy();
    void validate();
    Context getContext() const;

    void markDirty();
    bool isDirty() const;

    void setPrimitiveCount(unsigned int  num_primitives);
    unsigned int getPrimitiveCount() const;

    void setBoundingBoxProgram(Program  program);
    Program getBoundingBoxProgram() const;

    void setIntersectionProgram(Program  program);
    Program getIntersectionProgram() const;

    Variable declareVariable (const std::string& name);
    Variable queryVariable   (const std::string& name) const;
    void     removeVariable  (Variable v);
    unsigned int getVariableCount() const;
    Variable getVariable     (unsigned int index) const;

    RTgeometry get();

  private:
    typedef RTgeometry api_t;
    virtual ~GeometryObj() {}
    RTgeometry m_geometry;
    GeometryObj(RTgeometry geometry) : m_geometry(geometry) {}
    friend class Handle<GeometryObj>;
  };


  //
-----------------------------------------------------------------------------


  class MaterialObj : public ScopedObj {
  public:
    void destroy();
    void validate();
    Context getContext() const;

    void setClosestHitProgram(unsigned int ray_type_index, Program  program);
    Program getClosestHitProgram(unsigned int ray_type_index) const;

    void setAnyHitProgram(unsigned int ray_type_index, Program  program);
    Program getAnyHitProgram(unsigned int ray_type_index) const;

    Variable declareVariable (const std::string& name);
    Variable queryVariable   (const std::string& name) const;
    void     removeVariable  (Variable v);
    unsigned int getVariableCount() const;
    Variable getVariable     (unsigned int index) const;

    RTmaterial get();
  private:
    typedef RTmaterial api_t;
    virtual ~MaterialObj() {}
    RTmaterial m_material;
    MaterialObj(RTmaterial material) : m_material(material) {}
    friend class Handle<MaterialObj>;
  };


  //
-----------------------------------------------------------------------------


  class TextureSamplerObj : public DestroyableObj {
  public:
    void destroy();
    void validate();
    Context getContext() const;

    void setMipLevelCount (unsigned int  num_mip_levels);
    unsigned int getMipLevelCount () const;

    void setArraySize(unsigned int  num_textures_in_array);
    unsigned int getArraySize() const;

    void setWrapMode(unsigned int dim, RTwrapmode wrapmode);
    RTwrapmode getWrapMode(unsigned int dim) const;

    void setFilteringModes(RTfiltermode  minification, RTfiltermode magnification, RTfiltermode  mipmapping);
    void getFilteringModes(RTfiltermode& minification, RTfiltermode& magnification, RTfiltermode& mipmapping) const;

    void setMaxAnisotropy(float value);
    float getMaxAnisotropy() const;

    void setReadMode(RTtexturereadmode  readmode);
    RTtexturereadmode getReadMode() const;

    void setIndexingMode(RTtextureindexmode  indexmode);
    RTtextureindexmode getIndexingMode() const;

    int getId() const;

    void setBuffer(unsigned int texture_array_idx, unsigned int mip_level, Buffer buffer);
    Buffer getBuffer(unsigned int texture_array_idx, unsigned int mip_level) const;

    RTtexturesampler get();

    void registerGLTexture();
    void unregisterGLTexture();

#ifdef _WIN32

    void registerD3D9Texture();
    void registerD3D10Texture();
    void registerD3D11Texture();

    void unregisterD3D9Texture();
    void unregisterD3D10Texture();
    void unregisterD3D11Texture();

#endif

  private:
    typedef RTtexturesampler api_t;
    virtual ~TextureSamplerObj() {}
    RTtexturesampler m_texturesampler;
    TextureSamplerObj(RTtexturesampler texturesampler) : m_texturesampler(texturesampler) {}
    friend class Handle<TextureSamplerObj>;
  };


  // ---------------------------------------------------------------------------


  class BufferObj : public DestroyableObj {
  public:
    void destroy();
    void validate();
    Context getContext() const;

    void setFormat    (RTformat format);
    RTformat getFormat() const;

    void setElementSize  (RTsize size_of_element);
    RTsize getElementSize() const;

    void getDevicePointer( unsigned int optix_device_number, CUdeviceptr * device_pointer );

    void setDevicePointer( unsigned int optix_device_number, CUdeviceptr device_pointer );

    void markDirty();

    void setSize(RTsize  width);
    void getSize(RTsize& width) const;
    void setSize(RTsize  width, RTsize  height);
    void getSize(RTsize& width, RTsize& height) const;
    void setSize(RTsize  width, RTsize  height, RTsize  depth);
    void getSize(RTsize& width, RTsize& height, RTsize& depth) const;

    void setSize(unsigned int dimensionality, const RTsize* dims);
    void getSize(unsigned int dimensionality,       RTsize* dims) const;

    unsigned int getDimensionality() const;

    unsigned int getGLBOId() const;

    void registerGLBuffer();
    void unregisterGLBuffer();

#ifdef _WIN32

    void registerD3D9Buffer();
    void registerD3D10Buffer();
    void registerD3D11Buffer();

    void unregisterD3D9Buffer();
    void unregisterD3D10Buffer();
    void unregisterD3D11Buffer();

    IDirect3DResource9* getD3D9Resource();
    ID3D10Resource* getD3D10Resource();
    ID3D11Resource* getD3D11Resource();

#endif

    void* map();
    void unmap();

    RTbuffer get();

  private:
    typedef RTbuffer api_t;
    virtual ~BufferObj() {}
    RTbuffer m_buffer;
    BufferObj(RTbuffer buffer) : m_buffer(buffer) {}
    friend class Handle<BufferObj>;
  };


  // ----------------------------------------------------------------------------


  inline void APIObj::checkError( RTresult code ) const
  {
    if( code != RT_SUCCESS) {
      RTcontext c = this->getContext()->get();
      throw Exception::makeException( code, c );
    }
  }

  inline void APIObj::checkError( RTresult code, Context context ) const
  {
    if( code != RT_SUCCESS) {
      RTcontext c = context->get();
      throw Exception::makeException( code, c );
    }
  }

  inline void APIObj::checkErrorNoGetContext( RTresult code ) const
  {
    if( code != RT_SUCCESS) {
      throw Exception::makeException( code, 0u );
    }
  }

  inline Context ContextObj::getContext() const
  {
    return Context::take( m_context );
  }

  inline void ContextObj::checkError(RTresult code) const
  {
    if( code != RT_SUCCESS && code != RT_TIMEOUT_CALLBACK )
      throw Exception::makeException( code, m_context );
  }

  inline unsigned int ContextObj::getDeviceCount()
  {
    unsigned int count;
    if( RTresult code = rtDeviceGetDeviceCount(&count) )
      throw Exception::makeException( code, 0 );

    return count;
  }

  inline std::string ContextObj::getDeviceName(int ordinal)
  {
    const RTsize max_string_size = 256;
    char name[max_string_size];
    if( RTresult code = rtDeviceGetAttribute(ordinal, RT_DEVICE_ATTRIBUTE_NAME,
                                              max_string_size, name) )
      throw Exception::makeException( code, 0 );
    return std::string(name);
  }

  inline void ContextObj::getDeviceAttribute(int ordinal, RTdeviceattribute
attrib, RTsize size, void* p)
  {
    if( RTresult code = rtDeviceGetAttribute(ordinal, attrib, size, p) )
      throw Exception::makeException( code, 0 );
  }

  inline Context ContextObj::create()
  {
    RTcontext c;
    if( RTresult code = rtContextCreate(&c) )
      throw Exception::makeException( code, 0 );

    return Context::take(c);
  }

  inline void ContextObj::destroy()
  {
    checkErrorNoGetContext( rtContextDestroy( m_context ) );
    m_context = 0;
  }

  inline void ContextObj::validate()
  {
    checkError( rtContextValidate( m_context ) );
  }

  inline Acceleration ContextObj::createAcceleration(const char* builder, const
 char* traverser)
  {
    RTacceleration acceleration;
    checkError( rtAccelerationCreate( m_context, &acceleration ) );
    checkError( rtAccelerationSetBuilder( acceleration, builder ) );
    checkError( rtAccelerationSetTraverser( acceleration, traverser ) );
    return Acceleration::take(acceleration);
  }


  inline Buffer ContextObj::createBuffer(unsigned int type)
  {
    RTbuffer buffer;
    checkError( rtBufferCreate( m_context, type, &buffer ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBuffer(unsigned int type, RTformat format)
  {
    RTbuffer buffer;
    checkError( rtBufferCreate( m_context, type, &buffer ) );
    checkError( rtBufferSetFormat( buffer, format ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBuffer(unsigned int type, RTformat format,
RTsize width)
  {
    RTbuffer buffer;
    checkError( rtBufferCreate( m_context, type, &buffer ) );
    checkError( rtBufferSetFormat( buffer, format ) );
    checkError( rtBufferSetSize1D( buffer, width ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBuffer(unsigned int type, RTformat format,
RTsize width, RTsize height)
  {
    RTbuffer buffer;
    checkError( rtBufferCreate( m_context, type, &buffer ) );
    checkError( rtBufferSetFormat( buffer, format ) );
    checkError( rtBufferSetSize2D( buffer, width, height ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBuffer(unsigned int type, RTformat format,
RTsize width, RTsize height, RTsize depth)
  {
    RTbuffer buffer;
    checkError( rtBufferCreate( m_context, type, &buffer ) );
    checkError( rtBufferSetFormat( buffer, format ) );
    checkError( rtBufferSetSize3D( buffer, width, height, depth ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBufferForCUDA(unsigned int type)
  {
    RTbuffer buffer;
    checkError( rtBufferCreateForCUDA( m_context, type, &buffer ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBufferForCUDA(unsigned int type, RTformat
format)
  {
    RTbuffer buffer;
    checkError( rtBufferCreateForCUDA( m_context, type, &buffer ) );
    checkError( rtBufferSetFormat( buffer, format ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBufferForCUDA(unsigned int type, RTformat
format, RTsize width)
  {
    RTbuffer buffer;
    checkError( rtBufferCreateForCUDA( m_context, type, &buffer ) );
    checkError( rtBufferSetFormat( buffer, format ) );
    checkError( rtBufferSetSize1D( buffer, width ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBufferForCUDA(unsigned int type, RTformat
format, RTsize width, RTsize height)
  {
    RTbuffer buffer;
    checkError( rtBufferCreateForCUDA( m_context, type, &buffer ) );
    checkError( rtBufferSetFormat( buffer, format ) );
    checkError( rtBufferSetSize2D( buffer, width, height ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBufferForCUDA(unsigned int type, RTformat
format, RTsize width, RTsize height, RTsize depth)
  {
    RTbuffer buffer;
    checkError( rtBufferCreateForCUDA( m_context, type, &buffer ) );
    checkError( rtBufferSetFormat( buffer, format ) );
    checkError( rtBufferSetSize3D( buffer, width, height, depth ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBufferFromGLBO(unsigned int type, unsigned
int vbo)
  {
    RTbuffer buffer;
    checkError( rtBufferCreateFromGLBO( m_context, type, vbo, &buffer ) );
    return Buffer::take(buffer);
  }

#ifdef _WIN32

  inline Buffer ContextObj::createBufferFromD3D9Resource(unsigned int type,
IDirect3DResource9 *pResource)
  {
    RTbuffer buffer;
    checkError( rtBufferCreateFromD3D9Resource( m_context, type, pResource, &
buffer ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBufferFromD3D10Resource(unsigned int type,
ID3D10Resource *pResource)
  {
    RTbuffer buffer;
    checkError( rtBufferCreateFromD3D10Resource( m_context, type, pResource, &
buffer ) );
    return Buffer::take(buffer);
  }

  inline Buffer ContextObj::createBufferFromD3D11Resource(unsigned int type,
ID3D11Resource *pResource)
  {
    RTbuffer buffer;
    checkError( rtBufferCreateFromD3D11Resource( m_context, type, pResource, &
buffer ) );
    return Buffer::take(buffer);
  }

  inline TextureSampler ContextObj::createTextureSamplerFromD3D9Resource(
IDirect3DResource9 *pResource)
  {
    RTtexturesampler textureSampler;
    checkError( rtTextureSamplerCreateFromD3D9Resource(m_context, pResource, &
textureSampler));
    return TextureSampler::take(textureSampler);
  }

  inline TextureSampler ContextObj::createTextureSamplerFromD3D10Resource(
ID3D10Resource *pResource)
  {
    RTtexturesampler textureSampler;
    checkError( rtTextureSamplerCreateFromD3D10Resource(m_context, pResource, &
textureSampler));
    return TextureSampler::take(textureSampler);
  }

  inline TextureSampler ContextObj::createTextureSamplerFromD3D11Resource(
ID3D11Resource *pResource)
  {
    RTtexturesampler textureSampler;
    checkError( rtTextureSamplerCreateFromD3D11Resource(m_context, pResource, &
textureSampler));
    return TextureSampler::take(textureSampler);
  }

  inline void ContextObj::setD3D9Device(IDirect3DDevice9* device)
  {
    checkError( rtContextSetD3D9Device( m_context, device ) );
  }

  inline void ContextObj::setD3D10Device(ID3D10Device* device)
  {
    checkError( rtContextSetD3D10Device( m_context, device ) );
  }

  inline void ContextObj::setD3D11Device(ID3D11Device* device)
  {
    checkError( rtContextSetD3D11Device( m_context, device ) );
  }

#endif

  inline TextureSampler ContextObj::createTextureSamplerFromGLImage(unsigned
int id, RTgltarget target)
  {
    RTtexturesampler textureSampler;
    checkError( rtTextureSamplerCreateFromGLImage(m_context, id, target, &
textureSampler));
    return TextureSampler::take(textureSampler);
  }

  inline Geometry ContextObj::createGeometry()
  {
    RTgeometry geometry;
    checkError( rtGeometryCreate( m_context, &geometry ) );
    return Geometry::take(geometry);
  }

  inline GeometryInstance ContextObj::createGeometryInstance()
  {
    RTgeometryinstance geometryinstance;
    checkError( rtGeometryInstanceCreate( m_context, &geometryinstance ) );
    return GeometryInstance::take(geometryinstance);
  }

  template<class Iterator>
    GeometryInstance ContextObj::createGeometryInstance( Geometry geometry,
Iterator matlbegin, Iterator matlend)
  {
    GeometryInstance result = createGeometryInstance();
    result->setGeometry( geometry );
    unsigned int count = 0;
    for( Iterator iter = matlbegin; iter != matlend; ++iter )
      ++count;
    result->setMaterialCount( count );
    unsigned int index = 0;
    for(Iterator iter = matlbegin; iter != matlend; ++iter, ++index )
      result->setMaterial( index, *iter );
    return result;
  }

  inline Group ContextObj::createGroup()
  {
    RTgroup group;
    checkError( rtGroupCreate( m_context, &group ) );
    return Group::take(group);
  }

  template<class Iterator>
```

```
    inline Group ContextObj::createGroup( Iterator childbegin, Iterator childend )
  {
    Group result = createGroup();
    unsigned int count = 0;
    for(Iterator iter = childbegin; iter != childend; ++iter )
      ++count;
    result->setChildCount( count );
    unsigned int index = 0;
    for(Iterator iter = childbegin; iter != childend; ++iter, ++index )
      result->setChild( index, *iter );
    return result;
  }

  inline GeometryGroup ContextObj::createGeometryGroup()
  {
    RTgeometrygroup gg;
    checkError( rtGeometryGroupCreate( m_context, &gg ) );
    return GeometryGroup::take( gg );
  }

  template<class Iterator>
  inline GeometryGroup ContextObj::createGeometryGroup( Iterator childbegin, Iterator childend )
  {
    GeometryGroup result = createGeometryGroup();
    unsigned int count = 0;
    for(Iterator iter = childbegin; iter != childend; ++iter )
      ++count;
    result->setChildCount( count );
    unsigned int index = 0;
    for(Iterator iter = childbegin; iter != childend; ++iter, ++index )
      result->setChild( index, *iter );
    return result;
  }

  inline Transform ContextObj::createTransform()
  {
    RTtransform t;
    checkError( rtTransformCreate( m_context, &t ) );
    return Transform::take( t );
  }

  inline Material ContextObj::createMaterial()
  {
    RTmaterial material;
    checkError( rtMaterialCreate( m_context, &material ) );
    return Material::take(material);
  }

  inline Program ContextObj::createProgramFromPTXFile( const std::string& filename, const std::string& program_name )
  {
    RTprogram program;
    checkError( rtProgramCreateFromPTXFile( m_context, filename.c_str(), program_name.c_str(), &program ) );
    return Program::take(program);
  }

  inline Program ContextObj::createProgramFromPTXString( const std::string& ptx, const std::string& program_name )
  {
    RTprogram program;
    checkError( rtProgramCreateFromPTXString( m_context, ptx.c_str(), program_name.c_str(), &program ) );
    return Program::take(program);
  }

  inline Selector ContextObj::createSelector()
  {
    RTselector selector;
    checkError( rtSelectorCreate( m_context, &selector ) );
    return Selector::take(selector);
  }

  inline TextureSampler ContextObj::createTextureSampler()
  {
    RTtexturesampler texturesampler;
    checkError( rtTextureSamplerCreate( m_context, &texturesampler ) );
    return TextureSampler::take(texturesampler);
  }

  inline std::string ContextObj::getErrorString( RTresult code ) const
  {
    const char* str;
    rtContextGetErrorString( m_context, code, &str);
    return std::string(str);
  }

  template<class Iterator> inline
    void ContextObj::setDevices(Iterator begin, Iterator end)
  {
    std::vector<int> devices;
    std::copy( begin, end, std::insert_iterator<std::vector<int> >( devices,
devices.begin() ) );
    checkError( rtContextSetDevices( m_context, static_cast<unsigned int>(
devices.size()), &devices[0]) );
  }

  inline std::vector<int> ContextObj::getEnabledDevices() const
  {
    // Initialize with the number of enabled devices
    std::vector<int> devices(getEnabledDeviceCount());
    checkError( rtContextGetDevices( m_context, &devices[0] ) );
    return devices;
  }

  inline unsigned int ContextObj::getEnabledDeviceCount() const
  {
    unsigned int num;
    checkError( rtContextGetDeviceCount( m_context, &num ) );
    return num;
  }

  inline int ContextObj::getMaxTextureCount() const
  {
    int tex_count;
    checkError( rtContextGetAttribute( m_context,
RT_CONTEXT_ATTRIBUTE_MAX_TEXTURE_COUNT, sizeof(tex_count), &tex_count) );
    return tex_count;
  }

  inline int ContextObj::getCPUNumThreads() const
  {
    int cpu_num_threads;
    checkError( rtContextGetAttribute( m_context,
RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS, sizeof(cpu_num_threads), &cpu_num_threads) );
    return cpu_num_threads;
  }

  inline RTsize ContextObj::getUsedHostMemory() const
  {
    RTsize used_mem;
    checkError( rtContextGetAttribute( m_context,
RT_CONTEXT_ATTRIBUTE_USED_HOST_MEMORY, sizeof(used_mem), &used_mem) );
    return used_mem;
  }

  inline int ContextObj::getGPUPagingActive() const
  {
    int gpu_paging_active;
    checkError( rtContextGetAttribute( m_context,
RT_CONTEXT_ATTRIBUTE_GPU_PAGING_ACTIVE, sizeof(gpu_paging_active), &gpu_paging_active) );
    return gpu_paging_active;
  }

  inline int ContextObj::getGPUPagingForcedOff() const
  {
    int gpu_paging_forced_off;
    checkError( rtContextGetAttribute( m_context, RT_CONTEXT_ATTRIBUTE_GPU_PAGING_FORCED_OFF, sizeof(gpu_paging_forced_off), &gpu_paging_forced_off) );
    return gpu_paging_forced_off;
  }

  inline RTsize ContextObj::getAvailableDeviceMemory(int ordinal) const
  {
    RTsize free_mem;
    checkError( rtContextGetAttribute( m_context,
                                       static_cast<RTcontextattribute>( RT_CONTEXT_ATTRIBUTE_AVAILABLE_DEVICE_MEMORY + ordinal),
                                       sizeof(free_mem), &free_mem) );
    return free_mem;
  }

  inline void ContextObj::setCPUNumThreads(int cpu_num_threads)
  {
    checkError( rtContextSetAttribute( m_context, RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS, sizeof(cpu_num_threads), &cpu_num_threads) );
  }

  inline void ContextObj::setGPUPagingForcedOff(int gpu_paging_forced_off)
  {
    checkError( rtContextSetAttribute( m_context, RT_CONTEXT_ATTRIBUTE_GPU_PAGING_FORCED_OFF, sizeof(gpu_paging_forced_off), &gpu_paging_forced_off) );
  }

  inline void ContextObj::setStackSize(RTsize  stack_size_bytes)
  {
    checkError(rtContextSetStackSize(m_context, stack_size_bytes) );
  }

  inline RTsize ContextObj::getStackSize() const
  {
    RTsize result;
    checkError( rtContextGetStackSize( m_context, &result ) );
    return result;
  }

  inline void ContextObj::setTimeoutCallback(RTtimeoutcallback callback, double min_polling_seconds)
  {
    checkError( rtContextSetTimeoutCallback( m_context, callback, min_polling_seconds ) );
  }

  inline void ContextObj::setEntryPointCount(unsigned int  num_entry_points)
  {
    checkError( rtContextSetEntryPointCount( m_context, num_entry_points ) );
  }

  inline unsigned int ContextObj::getEntryPointCount() const
  {
    unsigned int result;
    checkError( rtContextGetEntryPointCount( m_context, &result ) );
    return result;
  }


  inline void ContextObj::setRayGenerationProgram(unsigned int entry_point_index, Program  program)
  {
    checkError( rtContextSetRayGenerationProgram( m_context, entry_point_index, program->get() ) );
  }

  inline Program ContextObj::getRayGenerationProgram(unsigned int entry_point_index) const
  {
    RTprogram result;
    checkError( rtContextGetRayGenerationProgram( m_context, entry_point_index, &result ) );
    return Program::take( result );
  }


  inline void ContextObj::setExceptionProgram(unsigned int entry_point_index, Program  program)
  {
    checkError( rtContextSetExceptionProgram( m_context, entry_point_index, program->get() ) );
  }

  inline Program ContextObj::getExceptionProgram(unsigned int entry_point_index) const
  {
    RTprogram result;
    checkError( rtContextGetExceptionProgram( m_context, entry_point_index, &result ) );
    return Program::take( result );
  }


  inline void ContextObj::setExceptionEnabled( RTexception exception, bool enabled )
  {
    checkError( rtContextSetExceptionEnabled( m_context, exception, enabled ) );
  }

  inline bool ContextObj::getExceptionEnabled( RTexception exception ) const
  {
    int enabled;
    checkError( rtContextGetExceptionEnabled( m_context, exception, &enabled ) );
    return enabled != 0;
  }


  inline void ContextObj::setRayTypeCount(unsigned int  num_ray_types)
  {
    checkError( rtContextSetRayTypeCount( m_context, num_ray_types ) );
  }

  inline unsigned int ContextObj::getRayTypeCount() const
  {
    unsigned int result;
    checkError( rtContextGetRayTypeCount( m_context, &result ) );
    return result;
  }

  inline void ContextObj::setMissProgram(unsigned int ray_type_index, Program program)
  {
    checkError( rtContextSetMissProgram( m_context, ray_type_index, program->get() ) );
  }

  inline Program ContextObj::getMissProgram(unsigned int ray_type_index) const
  {
    RTprogram result;
    checkError( rtContextGetMissProgram( m_context, ray_type_index, &result ) );
    return Program::take( result );
  }

  inline void ContextObj::compile()
  {
    checkError( rtContextCompile( m_context ) );
  }

  inline void ContextObj::launch(unsigned int entry_point_index, RTsize image_width)
  {
    checkError( rtContextLaunch1D( m_context, entry_point_index, image_width ) );
  }

  inline void ContextObj::launch(unsigned int entry_point_index, RTsize
image_width, RTsize image_height)
  {
    checkError( rtContextLaunch2D( m_context, entry_point_index, image_width,
image_height ) );
  }

  inline void ContextObj::launch(unsigned int entry_point_index, RTsize
image_width, RTsize image_height, RTsize image_depth)
  {
    checkError( rtContextLaunch3D( m_context, entry_point_index, image_width,
image_height, image_depth ) );
  }


  inline int ContextObj::getRunningState() const
  {
    int result;
    checkError( rtContextGetRunningState( m_context, &result ) );
    return result;
  }

  inline void ContextObj::setPrintEnabled(bool enabled)
  {
    checkError( rtContextSetPrintEnabled( m_context, enabled ) );
  }

  inline bool ContextObj::getPrintEnabled() const
  {
    int enabled;
    checkError( rtContextGetPrintEnabled( m_context, &enabled ) );
    return enabled != 0;
  }

  inline void ContextObj::setPrintBufferSize(RTsize buffer_size_bytes)
  {
    checkError( rtContextSetPrintBufferSize( m_context, buffer_size_bytes ) );
  }

  inline RTsize ContextObj::getPrintBufferSize() const
  {
    RTsize result;
    checkError( rtContextGetPrintBufferSize( m_context, &result ) );
    return result;
  }

  inline void ContextObj::setPrintLaunchIndex(int x, int y, int z)
  {
    checkError( rtContextSetPrintLaunchIndex( m_context, x, y, z ) );
  }

  inline optix::int3 ContextObj::getPrintLaunchIndex() const
  {
    optix::int3 result;
    checkError( rtContextGetPrintLaunchIndex( m_context, &result.x, &result.y,
&result.z ) );
    return result;
  }

  inline Variable ContextObj::declareVariable(const std::string& name)
  {
    RTvariable v;
    checkError( rtContextDeclareVariable( m_context, name.c_str(), &v ) );
    return Variable::take( v );
  }

  inline Variable ContextObj::queryVariable(const std::string& name) const
  {
    RTvariable v;
    checkError( rtContextQueryVariable( m_context, name.c_str(), &v ) );
    return Variable::take( v );
  }

```

```
inline void ContextObj::removeVariable(Variable v)
{
  checkError( rtContextRemoveVariable( m_context, v->get() ) );
}

inline unsigned int ContextObj::getVariableCount() const
{
  unsigned int result;
  checkError( rtContextGetVariableCount( m_context, &result ) );
  return result;
}

inline Variable ContextObj::getVariable(unsigned int index) const
{
  RTvariable v;
  checkError( rtContextGetVariable( m_context, index, &v ) );
  return Variable::take( v );
}


inline RTcontext ContextObj::get()
{
  return m_context;
}

inline void ProgramObj::destroy()
{
  Context context = getContext();
  checkError( rtProgramDestroy( m_program ), context );
  m_program = 0;
}

inline void ProgramObj::validate()
{
  checkError( rtProgramValidate( m_program ) );
}

inline Context ProgramObj::getContext() const
{
  RTcontext c;
  checkErrorNoGetContext( rtProgramGetContext( m_program, &c ) );
  return Context::take( c );
}

inline Variable ProgramObj::declareVariable(const std::string& name)
{
  RTvariable v;
  checkError( rtProgramDeclareVariable( m_program, name.c_str(), &v ) );
  return Variable::take( v );
}

inline Variable ProgramObj::queryVariable(const std::string& name) const
{
  RTvariable v;
  checkError( rtProgramQueryVariable( m_program, name.c_str(), &v ) );
  return Variable::take( v );
}

inline void ProgramObj::removeVariable(Variable v)
{
  checkError( rtProgramRemoveVariable( m_program, v->get() ) );
}

inline unsigned int ProgramObj::getVariableCount() const
{
  unsigned int result;
  checkError( rtProgramGetVariableCount( m_program, &result ) );
  return result;
}

inline Variable ProgramObj::getVariable(unsigned int index) const
{
  RTvariable v;
  checkError( rtProgramGetVariable( m_program, index, &v ) );
  return Variable::take(v);
}

inline RTprogram ProgramObj::get()
{
  return m_program;
}

inline void GroupObj::destroy()
{
  Context context = getContext();
  checkError( rtGroupDestroy( m_group ), context );
  m_group = 0;
}

inline void GroupObj::validate()
{
  checkError( rtGroupValidate( m_group ) );
}

inline Context GroupObj::getContext() const
{
  RTcontext c;
  checkErrorNoGetContext( rtGroupGetContext( m_group, &c) );
  return Context::take(c);
}

inline void SelectorObj::destroy()
{
  Context context = getContext();
  checkError( rtSelectorDestroy( m_selector ), context );
  m_selector = 0;
}

inline void SelectorObj::validate()
{
  checkError( rtSelectorValidate( m_selector ) );
}

inline Context SelectorObj::getContext() const
{
  RTcontext c;
  checkErrorNoGetContext( rtSelectorGetContext( m_selector, &c ) );
  return Context::take( c );
}

inline void SelectorObj::setVisitProgram(Program program)
{
  checkError( rtSelectorSetVisitProgram( m_selector, program->get() ) );
}

inline Program SelectorObj::getVisitProgram() const
{
  RTprogram result;
  checkError( rtSelectorGetVisitProgram( m_selector, &result ) );
  return Program::take( result );
}

inline void SelectorObj::setChildCount(unsigned int count)
{
  checkError( rtSelectorSetChildCount( m_selector, count) );
}

inline unsigned int SelectorObj::getChildCount() const
{
  unsigned int result;
  checkError( rtSelectorGetChildCount( m_selector, &result ) );
  return result;
}

template< typename T >
inline void SelectorObj::setChild(unsigned int index, T child)
{
  checkError( rtSelectorSetChild( m_selector, index, child->get() ) );
```

```
}

template< typename T >
inline T SelectorObj::getChild(unsigned int index) const
{
  RTobject result;
  checkError( rtSelectorGetChild( m_selector, index, &result ) );
  return T::take( result );
}

inline Variable SelectorObj::declareVariable(const std::string& name)
{
  RTvariable v;
  checkError( rtSelectorDeclareVariable( m_selector, name.c_str(), &v ) );
  return Variable::take( v );
}

inline Variable SelectorObj::queryVariable(const std::string& name) const
{
  RTvariable v;
  checkError( rtSelectorQueryVariable( m_selector, name.c_str(), &v ) );
  return Variable::take( v );
}

inline void SelectorObj::removeVariable(Variable v)
{
  checkError( rtSelectorRemoveVariable( m_selector, v->get() ) );
}

inline unsigned int SelectorObj::getVariableCount() const
{
  unsigned int result;
  checkError( rtSelectorGetVariableCount( m_selector, &result ) );
  return result;
}

inline Variable SelectorObj::getVariable(unsigned int index) const
{
  RTvariable v;
  checkError( rtSelectorGetVariable( m_selector, index, &v ) );
  return Variable::take( v );
}

inline RTselector SelectorObj::get()
{
  return m_selector;
}

inline void GroupObj::setAcceleration(Acceleration acceleration)
{
  checkError( rtGroupSetAcceleration( m_group, acceleration->get() ) );
}

inline Acceleration GroupObj::getAcceleration() const
{
  RTacceleration result;
  checkError( rtGroupGetAcceleration( m_group, &result ) );
  return Acceleration::take( result );
}

inline void GroupObj::setChildCount(unsigned int  count)
{
  checkError( rtGroupSetChildCount( m_group, count ) );
}

inline unsigned int GroupObj::getChildCount() const
{
  unsigned int result;
  checkError( rtGroupGetChildCount( m_group, &result ) );
  return result;
}

template< typename T >
inline void GroupObj::setChild(unsigned int index, T child)
```

```
  {
    checkError( rtGroupSetChild( m_group, index, child->get() ) );
  }

  template< typename T >
  inline T GroupObj::getChild(unsigned int index) const
  {
    RTobject result;
    checkError( rtGroupGetChild( m_group, index, &result) );
    return T::take( result );
  }

  inline RTgroup GroupObj::get()
  {
    return m_group;
  }

  inline void GeometryGroupObj::destroy()
  {
    Context context = getContext();
    checkError( rtGeometryGroupDestroy( m_geometrygroup ), context );
    m_geometrygroup = 0;
  }

  inline void GeometryGroupObj::validate()
  {
    checkError( rtGeometryGroupValidate( m_geometrygroup ) );
  }

  inline Context GeometryGroupObj::getContext() const
  {
    RTcontext c;
    checkErrorNoGetContext( rtGeometryGroupGetContext( m_geometrygroup, &c) );
    return Context::take(c);
  }

  inline void GeometryGroupObj::setAcceleration(Acceleration acceleration)
  {
    checkError( rtGeometryGroupSetAcceleration( m_geometrygroup, acceleration->
get() ) );
  }

  inline Acceleration GeometryGroupObj::getAcceleration() const
  {
    RTacceleration result;
    checkError( rtGeometryGroupGetAcceleration( m_geometrygroup, &result ) );
    return Acceleration::take( result );
  }

  inline void GeometryGroupObj::setChildCount(unsigned int  count)
  {
    checkError( rtGeometryGroupSetChildCount( m_geometrygroup, count ) );
  }

  inline unsigned int GeometryGroupObj::getChildCount() const
  {
    unsigned int result;
    checkError( rtGeometryGroupGetChildCount( m_geometrygroup, &result ) );
    return result;
  }

  inline void GeometryGroupObj::setChild(unsigned int index, GeometryInstance
child)
  {
    checkError( rtGeometryGroupSetChild( m_geometrygroup, index, child->get() )
 );
  }

  inline GeometryInstance GeometryGroupObj::getChild(unsigned int index) const
  {
    RTgeometryinstance result;
    checkError( rtGeometryGroupGetChild( m_geometrygroup, index, &result) );
    return GeometryInstance::take( result );
  }

  inline RTgeometrygroup GeometryGroupObj::get()
  {
    return m_geometrygroup;
  }

  inline void TransformObj::destroy()
  {
    Context context = getContext();
    checkError( rtTransformDestroy( m_transform ), context );
    m_transform = 0;
  }

  inline void TransformObj::validate()
  {
    checkError( rtTransformValidate( m_transform ) );
  }

  inline Context TransformObj::getContext() const
  {
    RTcontext c;
    checkErrorNoGetContext( rtTransformGetContext( m_transform, &c) );
    return Context::take(c);
  }

  template< typename T >
  inline void TransformObj::setChild(T child)
  {
    checkError( rtTransformSetChild( m_transform, child->get() ) );
  }

  template< typename T >
  inline T TransformObj::getChild() const
  {
    RTobject result;
    checkError( rtTransformGetChild( m_transform, &result) );
    return T::take( result );
  }

  inline void TransformObj::setMatrix(bool transpose, const float* matrix,
const float* inverse_matrix)
  {
    rtTransformSetMatrix( m_transform, transpose, matrix, inverse_matrix );
  }

  inline void TransformObj::getMatrix(bool transpose, float* matrix, float*
inverse_matrix) const
  {
    rtTransformGetMatrix( m_transform, transpose, matrix, inverse_matrix );
  }

  inline RTtransform TransformObj::get()
  {
    return m_transform;
  }

  inline void AccelerationObj::destroy()
  {
    Context context = getContext();
    checkError( rtAccelerationDestroy(m_acceleration), context );
    m_acceleration = 0;
  }

  inline void AccelerationObj::validate()
  {
    checkError( rtAccelerationValidate(m_acceleration) );
  }

  inline Context AccelerationObj::getContext() const
  {
    RTcontext c;
    checkErrorNoGetContext( rtAccelerationGetContext(m_acceleration, &c ) );
    return Context::take( c );
  }

  inline void AccelerationObj::markDirty()
  {
    checkError( rtAccelerationMarkDirty(m_acceleration) );
  }

  inline bool AccelerationObj::isDirty() const
  {
    int dirty;
    checkError( rtAccelerationIsDirty(m_acceleration,&dirty) );
    return dirty != 0;
  }

  inline void AccelerationObj::setProperty( const std::string& name, const
std::string& value )
  {
    checkError( rtAccelerationSetProperty(m_acceleration, name.c_str(), value.
c_str() ) );
  }

  inline std::string AccelerationObj::getProperty( const std::string& name )
 const
  {
    const char* s;
    checkError( rtAccelerationGetProperty(m_acceleration, name.c_str(), &s ) );
    return std::string( s );
  }

  inline void AccelerationObj::setBuilder(const std::string& builder)
  {
    checkError( rtAccelerationSetBuilder(m_acceleration, builder.c_str() ) );
  }

  inline std::string AccelerationObj::getBuilder() const
  {
    const char* s;
    checkError( rtAccelerationGetBuilder(m_acceleration, &s ) );
    return std::string( s );
  }

  inline void AccelerationObj::setTraverser(const std::string& traverser)
  {
    checkError( rtAccelerationSetTraverser(m_acceleration, traverser.c_str() )
);
  }

  inline std::string AccelerationObj::getTraverser() const
  {
    const char* s;
    checkError( rtAccelerationGetTraverser(m_acceleration, &s ) );
    return std::string( s );
  }

  inline RTsize AccelerationObj::getDataSize() const
  {
    RTsize sz;
    checkError( rtAccelerationGetDataSize(m_acceleration, &sz) );
    return sz;
  }

  inline void AccelerationObj::getData( void* data ) const
  {
    checkError( rtAccelerationGetData(m_acceleration,data) );
  }

  inline void AccelerationObj::setData( const void* data, RTsize size )
  {
    checkError( rtAccelerationSetData(m_acceleration,data,size) );
  }

  inline RTacceleration AccelerationObj::get()
  {
    return m_acceleration;
  }

  inline void GeometryInstanceObj::destroy()
  {
    Context context = getContext();
    checkError( rtGeometryInstanceDestroy( m_geometryinstance ), context );
    m_geometryinstance = 0;
  }

  inline void GeometryInstanceObj::validate()
  {
    checkError( rtGeometryInstanceValidate( m_geometryinstance ) );
  }

  inline Context GeometryInstanceObj::getContext() const
  {
    RTcontext c;
    checkErrorNoGetContext( rtGeometryInstanceGetContext( m_geometryinstance, &
c ) );
    return Context::take( c );
  }

  inline void GeometryInstanceObj::setGeometry(Geometry geometry)
  {
    checkError( rtGeometryInstanceSetGeometry( m_geometryinstance, geometry->get
() ) );
  }

  inline Geometry GeometryInstanceObj::getGeometry() const
  {
    RTgeometry result;
    checkError( rtGeometryInstanceGetGeometry( m_geometryinstance, &result ) );
    return Geometry::take( result );
  }

  inline void GeometryInstanceObj::setMaterialCount(unsigned int  count)
  {
    checkError( rtGeometryInstanceSetMaterialCount( m_geometryinstance, count )
 );
  }

  inline unsigned int GeometryInstanceObj::getMaterialCount() const
  {
    unsigned int result;
    checkError( rtGeometryInstanceGetMaterialCount( m_geometryinstance, &result
 ) );
    return result;
  }

  inline void GeometryInstanceObj::setMaterial(unsigned int idx, Material
material)
  {
    checkError( rtGeometryInstanceSetMaterial( m_geometryinstance, idx,
material->get()) );
  }

  inline Material GeometryInstanceObj::getMaterial(unsigned int idx) const
  {
    RTmaterial result;
    checkError( rtGeometryInstanceGetMaterial( m_geometryinstance, idx, &result
 ) );
    return Material::take( result );
  }

  // Adds the material and returns the index to the added material.
  inline unsigned int GeometryInstanceObj::addMaterial(Material material)
  {
    unsigned int old_count = getMaterialCount();
    setMaterialCount(old_count+1);
    setMaterial(old_count, material);
    return old_count;
  }

  inline Variable GeometryInstanceObj::declareVariable(const std::string& name)
  {
    RTvariable v;
    checkError( rtGeometryInstanceDeclareVariable( m_geometryinstance, name.
c_str(), &v ) );
    return Variable::take( v );
  }

  inline Variable GeometryInstanceObj::queryVariable(const std::string& name)
 const
  {
    RTvariable v;
    checkError( rtGeometryInstanceQueryVariable( m_geometryinstance, name.c_str
(), &v ) );
    return Variable::take( v );
  }

  inline void GeometryInstanceObj::removeVariable(Variable v)
  {
    checkError( rtGeometryInstanceRemoveVariable( m_geometryinstance, v->get()
) );
  }

  inline unsigned int GeometryInstanceObj::getVariableCount() const
  {
    unsigned int result;
    checkError( rtGeometryInstanceGetVariableCount( m_geometryinstance, &result
 ) );
    return result;
  }

  inline Variable GeometryInstanceObj::getVariable(unsigned int index) const
  {
    RTvariable v;
    checkError( rtGeometryInstanceGetVariable( m_geometryinstance, index, &v )
);
    return Variable::take( v );
  }

  inline RTgeometryinstance GeometryInstanceObj::get()
  {
    return m_geometryinstance;
  }

  inline void GeometryObj::destroy()
  {
    Context context = getContext();
    checkError( rtGeometryDestroy( m_geometry ), context );
    m_geometry = 0;
  }

  inline void GeometryObj::validate()
  {
    checkError( rtGeometryValidate( m_geometry ) );
  }

  inline Context GeometryObj::getContext() const
  {
    RTcontext c;
    checkErrorNoGetContext( rtGeometryGetContext( m_geometry, &c ) );
    return Context::take( c );
  }

  inline void GeometryObj::setPrimitiveCount(unsigned int  num_primitives)
  {
    checkError( rtGeometrySetPrimitiveCount( m_geometry, num_primitives ) );
  }

  inline unsigned int GeometryObj::getPrimitiveCount() const
  {
    unsigned int result;
    checkError( rtGeometryGetPrimitiveCount( m_geometry, &result ) );
    return result;
  }

  inline void GeometryObj::setBoundingBoxProgram(Program  program)
  {
    checkError( rtGeometrySetBoundingBoxProgram( m_geometry, program->get() ) )
;
  }

  inline Program GeometryObj::getBoundingBoxProgram() const
  {
    RTprogram result;
    checkError( rtGeometryGetBoundingBoxProgram( m_geometry, &result ) );
    return Program::take( result );
  }

  inline void GeometryObj::setIntersectionProgram(Program  program)
  {
    checkError( rtGeometrySetIntersectionProgram( m_geometry, program->get() )
);
  }

  inline Program GeometryObj::getIntersectionProgram() const
  {
    RTprogram result;
    checkError( rtGeometryGetIntersectionProgram( m_geometry, &result ) );
    return Program::take( result );
  }

  inline Variable GeometryObj::declareVariable(const std::string& name)
  {
    RTvariable v;
    checkError( rtGeometryDeclareVariable( m_geometry, name.c_str(), &v ) );
    return Variable::take( v );
  }

  inline Variable GeometryObj::queryVariable(const std::string& name) const
  {
    RTvariable v;
    checkError( rtGeometryQueryVariable( m_geometry, name.c_str(), &v ) );
    return Variable::take( v );
  }

  inline void GeometryObj::removeVariable(Variable v)
  {
    checkError( rtGeometryRemoveVariable( m_geometry, v->get() ) );
  }

  inline unsigned int GeometryObj::getVariableCount() const
  {
    unsigned int result;
    checkError( rtGeometryGetVariableCount( m_geometry, &result ) );
    return result;
  }

  inline Variable GeometryObj::getVariable(unsigned int index) const
  {
    RTvariable v;
    checkError( rtGeometryGetVariable( m_geometry, index, &v ) );
    return Variable::take( v );
  }

  inline void GeometryObj::markDirty()
  {
    checkError( rtGeometryMarkDirty(m_geometry) );
  }

  inline bool GeometryObj::isDirty() const
  {
    int dirty;
    checkError( rtGeometryIsDirty(m_geometry,&dirty) );
    return dirty != 0;
  }

  inline RTgeometry GeometryObj::get()
  {
    return m_geometry;
  }

  inline void MaterialObj::destroy()
  {
    Context context = getContext();
    checkError( rtMaterialDestroy( m_material ), context );
    m_material = 0;
  }

  inline void MaterialObj::validate()
  {
    checkError( rtMaterialValidate( m_material ) );
  }

  inline Context MaterialObj::getContext() const
  {
    RTcontext c;
    checkErrorNoGetContext( rtMaterialGetContext( m_material, &c ) );
    return Context::take( c );
  }

  inline void MaterialObj::setClosestHitProgram(unsigned int ray_type_index,
Program  program)
  {
    checkError( rtMaterialSetClosestHitProgram( m_material, ray_type_index,
program->get() ) );
  }

  inline Program MaterialObj::getClosestHitProgram(unsigned int ray_type_index)
 const
  {
    RTprogram result;
    checkError( rtMaterialGetClosestHitProgram( m_material, ray_type_index, &
result ) );
    return Program::take( result );
  }

  inline void MaterialObj::setAnyHitProgram(unsigned int ray_type_index, Program
  program)
  {
    checkError( rtMaterialSetAnyHitProgram( m_material, ray_type_index, program
->get() ) );
  }

  inline Program MaterialObj::getAnyHitProgram(unsigned int ray_type_index)
 const
  {
    RTprogram result;
    checkError( rtMaterialGetAnyHitProgram( m_material, ray_type_index, &result
 ) );
    return Program::take( result );
  }

  inline Variable MaterialObj::declareVariable(const std::string& name)
  {
    RTvariable v;
    checkError( rtMaterialDeclareVariable( m_material, name.c_str(), &v ) );
    return Variable::take( v );
  }

  inline Variable MaterialObj::queryVariable(const std::string& name) const
  {
    RTvariable v;
    checkError( rtMaterialQueryVariable( m_material, name.c_str(), &v) );
    return Variable::take( v );
  }

  inline void MaterialObj::removeVariable(Variable v)
  {
    checkError( rtMaterialRemoveVariable( m_material, v->get() ) );
  }

  inline unsigned int MaterialObj::getVariableCount() const
  {
    unsigned int result;
    checkError( rtMaterialGetVariableCount( m_material, &result ) );
    return result;
  }

  inline Variable MaterialObj::getVariable(unsigned int index) const
  {
    RTvariable v;
    checkError( rtMaterialGetVariable( m_material, index, &v) );
    return Variable::take( v );
  }

  inline RTmaterial MaterialObj::get()
  {
    return m_material;
  }

  inline void TextureSamplerObj::destroy()
  {
    Context context = getContext();
    checkError( rtTextureSamplerDestroy( m_texturesampler ), context );
    m_texturesampler = 0;
  }

  inline void TextureSamplerObj::validate()
  {
    checkError( rtTextureSamplerValidate( m_texturesampler ) );
  }

  inline Context TextureSamplerObj::getContext() const
  {
    RTcontext c;
    checkErrorNoGetContext( rtTextureSamplerGetContext( m_texturesampler, &c )
);
    return Context::take( c );
  }

  inline void TextureSamplerObj::setMipLevelCount(unsigned int  num_mip_levels)
  {
    checkError( rtTextureSamplerSetMipLevelCount(m_texturesampler,
num_mip_levels ) );
  }

  inline unsigned int TextureSamplerObj::getMipLevelCount() const
  {
    unsigned int result;
    checkError( rtTextureSamplerGetMipLevelCount( m_texturesampler, &result ) )
;
    return result;
  }

  inline void TextureSamplerObj::setArraySize(unsigned int
num_textures_in_array)
  {
    checkError( rtTextureSamplerSetArraySize( m_texturesampler,
num_textures_in_array ) );
  }

  inline unsigned int TextureSamplerObj::getArraySize() const
  {
    unsigned int result;
    checkError( rtTextureSamplerGetArraySize( m_texturesampler, &result ) );
    return result;
  }

  inline void TextureSamplerObj::setWrapMode(unsigned int dim, RTwrapmode
wrapmode)
  {
    checkError( rtTextureSamplerSetWrapMode( m_texturesampler, dim, wrapmode )
);
  }

  inline RTwrapmode TextureSamplerObj::getWrapMode(unsigned int dim) const
  {
    RTwrapmode wrapmode;
    checkError( rtTextureSamplerGetWrapMode( m_texturesampler, dim, &wrapmode )
 );
    return wrapmode;
  }

  inline void TextureSamplerObj::setFilteringModes(RTfiltermode  minification,
RTfiltermode  magnification, RTfiltermode  mipmapping)
  {
    checkError( rtTextureSamplerSetFilteringModes( m_texturesampler,
minification, magnification, mipmapping ) );
  }

  inline void TextureSamplerObj::getFilteringModes(RTfiltermode& minification,
RTfiltermode& magnification, RTfiltermode& mipmapping) const
  {
    checkError( rtTextureSamplerGetFilteringModes( m_texturesampler, &
minification, &magnification, &mipmapping ) );
  }

  inline void TextureSamplerObj::setMaxAnisotropy(float value)
  {
    checkError( rtTextureSamplerSetMaxAnisotropy(m_texturesampler, value ) );
  }

  inline float TextureSamplerObj::getMaxAnisotropy() const
  {
    float result;
    checkError( rtTextureSamplerGetMaxAnisotropy( m_texturesampler, &result) );
    return result;
  }

  inline int TextureSamplerObj::getId() const
  {
    int result;
    checkError( rtTextureSamplerGetId( m_texturesampler, &result) );
    return result;
  }

  inline void TextureSamplerObj::setReadMode(RTtexturereadmode  readmode)
  {
    checkError( rtTextureSamplerSetReadMode( m_texturesampler, readmode ) );
  }

  inline RTtexturereadmode TextureSamplerObj::getReadMode() const
  {
    RTtexturereadmode result;
    checkError( rtTextureSamplerGetReadMode( m_texturesampler, &result) );
    return result;
  }

  inline void TextureSamplerObj::setIndexingMode(RTtextureindexmode  indexmode)
  {
    checkError( rtTextureSamplerSetIndexingMode( m_texturesampler, indexmode )
);
  }

  inline RTtextureindexmode TextureSamplerObj::getIndexingMode() const
  {
    RTtextureindexmode result;
    checkError( rtTextureSamplerGetIndexingMode( m_texturesampler, &result ) );
    return result;
  }

  inline void TextureSamplerObj::setBuffer(unsigned int texture_array_idx,
unsigned int mip_level, Buffer buffer)
  {
    checkError( rtTextureSamplerSetBuffer( m_texturesampler, texture_array_idx,
 mip_level, buffer->get() ) );
  }

  inline Buffer TextureSamplerObj::getBuffer(unsigned int texture_array_idx,
unsigned int mip_level) const
  {
    RTbuffer result;
    checkError( rtTextureSamplerGetBuffer(m_texturesampler, texture_array_idx,
 mip_level, &result ) );
    return Buffer::take(result);
  }

  inline RTtexturesampler TextureSamplerObj::get()
  {
    return m_texturesampler;
  }

  inline void TextureSamplerObj::registerGLTexture()
  {
    checkError( rtTextureSamplerGLRegister( m_texturesampler ) );
  }

  inline void TextureSamplerObj::unregisterGLTexture()
  {
    checkError( rtTextureSamplerGLUnregister( m_texturesampler ) );
  }

#ifdef _WIN32

  inline void TextureSamplerObj::registerD3D9Texture()
  {
    checkError( rtTextureSamplerD3D9Register( m_texturesampler ) );
  }

  inline void TextureSamplerObj::registerD3D10Texture()
  {
    checkError( rtTextureSamplerD3D10Register( m_texturesampler ) );
  }

  inline void TextureSamplerObj::registerD3D11Texture()
  {
    checkError( rtTextureSamplerD3D11Register( m_texturesampler ) );
  }

  inline void TextureSamplerObj::unregisterD3D9Texture()
  {
    checkError( rtTextureSamplerD3D9Unregister( m_texturesampler ) );
  }

  inline void TextureSamplerObj::unregisterD3D10Texture()
  {
    checkError( rtTextureSamplerD3D10Unregister( m_texturesampler ) );
  }

  inline void TextureSamplerObj::unregisterD3D11Texture()
  {
    checkError( rtTextureSamplerD3D11Unregister( m_texturesampler ) );
  }

#endif

  inline void BufferObj::destroy()
  {
    Context context = getContext();
    checkError( rtBufferDestroy( m_buffer ), context );
    m_buffer = 0;
  }

  inline void BufferObj::validate()
  {
    checkError( rtBufferValidate( m_buffer ) );
  }

  inline Context BufferObj::getContext() const
  {
    RTcontext c;
    checkErrorNoGetContext( rtBufferGetContext( m_buffer, &c ) );
    return Context::take( c );
  }

  inline void BufferObj::setFormat(RTformat format)
  {
    checkError( rtBufferSetFormat( m_buffer, format ) );
  }

  inline RTformat BufferObj::getFormat() const
  {
    RTformat result;
    checkError( rtBufferGetFormat( m_buffer, &result ) );
    return result;
  }

  inline void BufferObj::setElementSize(RTsize size_of_element)
  {
    checkError( rtBufferSetElementSize ( m_buffer, size_of_element ) );
  }

  inline RTsize BufferObj::getElementSize() const
  {
    RTsize result;
    checkError( rtBufferGetElementSize ( m_buffer, &result) );
    return result;
  }

  inline void BufferObj::getDevicePointer(unsigned int optix_device_number,
CUdeviceptr *device_pointer)
  {
    checkError( rtBufferGetDevicePointer( m_buffer, optix_device_number, (void*
*)device_pointer ) );
  }

  inline void BufferObj::setDevicePointer(unsigned int optix_device_number,
CUdeviceptr device_pointer)
  {
    checkError( rtBufferSetDevicePointer( m_buffer, optix_device_number,
device_pointer ) );
  }

  inline void BufferObj::markDirty()
  {
    checkError( rtBufferMarkDirty( m_buffer ) );
  }

  inline void BufferObj::setSize(RTsize width)
  {
    checkError( rtBufferSetSize1D( m_buffer, width ) );
  }

  inline void BufferObj::getSize(RTsize& width) const
  {
    checkError( rtBufferGetSize1D( m_buffer, &width ) );
  }

  inline void BufferObj::setSize(RTsize width, RTsize height)
  {
    checkError( rtBufferSetSize2D( m_buffer, width, height ) );
  }

  inline void BufferObj::getSize(RTsize& width, RTsize& height) const
  {
    checkError( rtBufferGetSize2D( m_buffer, &width, &height ) );
  }

  inline void BufferObj::setSize(RTsize width, RTsize height, RTsize depth)
  {
    checkError( rtBufferSetSize3D( m_buffer, width, height, depth ) );
  }

  inline void BufferObj::getSize(RTsize& width, RTsize& height, RTsize& depth)
 const
  {
    checkError( rtBufferGetSize3D( m_buffer, &width, &height, &depth ) );
  }

  inline void BufferObj::setSize(unsigned int dimensionality, const RTsize*
      dims)
  {
    checkError( rtBufferSetSizev( m_buffer, dimensionality, dims ) );
  }

  inline void BufferObj::getSize(unsigned int dimensionality, RTsize* dims)
 const
  {
    checkError( rtBufferGetSizev( m_buffer, dimensionality, dims ) );
  }

  inline unsigned int BufferObj::getDimensionality() const
  {
    unsigned int result;
    checkError( rtBufferGetDimensionality( m_buffer, &result ) );
    return result;
  }

  inline unsigned int BufferObj::getGLBOId() const
  {
    unsigned int result;
    checkError( rtBufferGetGLBOId( m_buffer, &result ) );
    return result;
  }

  inline void BufferObj::registerGLBuffer()
  {
    checkError( rtBufferGLRegister( m_buffer ) );
  }

  inline void BufferObj::unregisterGLBuffer()
  {
    checkError( rtBufferGLUnregister( m_buffer ) );
  }

#ifdef _WIN32

  inline void BufferObj::registerD3D9Buffer()
  {
    checkError( rtBufferD3D9Register( m_buffer ) );
  }

  inline void BufferObj::registerD3D10Buffer()
  {
    checkError( rtBufferD3D10Register( m_buffer ) );
  }

  inline void BufferObj::registerD3D11Buffer()
  {
    checkError( rtBufferD3D11Register( m_buffer ) );
  }

  inline void BufferObj::unregisterD3D9Buffer()
  {
    checkError( rtBufferD3D9Unregister( m_buffer ) );
  }

  inline void BufferObj::unregisterD3D10Buffer()
  {
    checkError( rtBufferD3D10Unregister( m_buffer ) );
  }

  inline void BufferObj::unregisterD3D11Buffer()
  {
    checkError( rtBufferD3D11Unregister( m_buffer ) );
  }

  inline IDirect3DResource9* BufferObj::getD3D9Resource()
  {
    IDirect3DResource9* result = NULL;
    checkError( rtBufferGetD3D9Resource( m_buffer, &result ) );
    return result;
  }

  inline ID3D10Resource* BufferObj::getD3D10Resource()
  {
    ID3D10Resource* result = NULL;
    checkError( rtBufferGetD3D10Resource( m_buffer, &result ) );
    return result;
  }

  inline ID3D11Resource* BufferObj::getD3D11Resource()
  {
    ID3D11Resource* result = NULL;
    checkError( rtBufferGetD3D11Resource( m_buffer, &result ) );
    return result;
  }

#endif

  inline void* BufferObj::map()
  {
    void* result;
    checkError( rtBufferMap( m_buffer, &result ) );
    return result;
  }

  inline void BufferObj::unmap()
  {
    checkError( rtBufferUnmap( m_buffer ) );
  }


  inline RTbuffer BufferObj::get()
  {
    return m_buffer;
  }

  inline Context VariableObj::getContext() const
  {
    RTcontext c;
    checkErrorNoGetContext( rtVariableGetContext( m_variable, &c ) );
    return Context::take( c );
  }

  inline void VariableObj::setUint(unsigned int u1)
  {
    checkError( rtVariableSet1ui( m_variable, u1 ) );
  }

  inline void VariableObj::setUint(unsigned int u1, unsigned int u2)
  {
    checkError( rtVariableSet2ui( m_variable, u1, u2 ) );
  }

  inline void VariableObj::setUint(unsigned int u1, unsigned int u2, unsigned
int u3)
  {
    checkError( rtVariableSet3ui( m_variable, u1, u2, u3 ) );
  }

  inline void VariableObj::setUint(unsigned int u1, unsigned int u2, unsigned
int u3, unsigned int u4)
  {
    checkError( rtVariableSet4ui( m_variable, u1, u2, u3, u4 ) );
  }

  inline void VariableObj::setUint(optix::uint2 u)
  {
    checkError( rtVariableSet2uiv( m_variable, &u.x ) );
  }

  inline void VariableObj::setUint(optix::uint3 u)
  {
    checkError( rtVariableSet3uiv( m_variable, &u.x ) );
  }

  inline void VariableObj::setUint(optix::uint4 u)
```

```
{
  checkError( rtVariableSet4uiv( m_variable, &u.x ) );
}

inline void VariableObj::set1uiv(const unsigned int* u)
{
  checkError( rtVariableSet1uiv( m_variable, u ) );
}

inline void VariableObj::set2uiv(const unsigned int* u)
{
  checkError( rtVariableSet2uiv( m_variable, u ) );
}

inline void VariableObj::set3uiv(const unsigned int* u)
{
  checkError( rtVariableSet3uiv( m_variable, u ) );
}

inline void VariableObj::set4uiv(const unsigned int* u)
{
  checkError( rtVariableSet4uiv( m_variable, u ) );
}

inline void VariableObj::setMatrix2x2fv(bool transpose, const float* m)
{
  checkError( rtVariableSetMatrix2x2fv( m_variable, (int)transpose, m ) );
}

inline void VariableObj::setMatrix2x3fv(bool transpose, const float* m)
{
  checkError( rtVariableSetMatrix2x3fv( m_variable, (int)transpose, m ) );
}

inline void VariableObj::setMatrix2x4fv(bool transpose, const float* m)
{
  checkError( rtVariableSetMatrix2x4fv( m_variable, (int)transpose, m ) );
}

inline void VariableObj::setMatrix3x2fv(bool transpose, const float* m)
{
  checkError( rtVariableSetMatrix3x2fv( m_variable, (int)transpose, m ) );
}

inline void VariableObj::setMatrix3x3fv(bool transpose, const float* m)
{
  checkError( rtVariableSetMatrix3x3fv( m_variable, (int)transpose, m ) );
}

inline void VariableObj::setMatrix3x4fv(bool transpose, const float* m)
{
  checkError( rtVariableSetMatrix3x4fv( m_variable, (int)transpose, m ) );
}

inline void VariableObj::setMatrix4x2fv(bool transpose, const float* m)
{
  checkError( rtVariableSetMatrix4x2fv( m_variable, (int)transpose, m ) );
}

inline void VariableObj::setMatrix4x3fv(bool transpose, const float* m)
{
  checkError( rtVariableSetMatrix4x3fv( m_variable, (int)transpose, m ) );
}

inline void VariableObj::setMatrix4x4fv(bool transpose, const float* m)
{
  checkError( rtVariableSetMatrix4x4fv( m_variable, (int)transpose, m ) );
}

inline void VariableObj::setFloat(float f1)
{
  checkError( rtVariableSet1f( m_variable, f1 ) );
}

```

```
inline void VariableObj::setFloat(optix::float2 f)
{
  checkError( rtVariableSet2fv( m_variable, &f.x ) );
}

inline void VariableObj::setFloat(float f1, float f2)
{
  checkError( rtVariableSet2f( m_variable, f1, f2 ) );
}

inline void VariableObj::setFloat(optix::float3 f)
{
  checkError( rtVariableSet3fv( m_variable, &f.x ) );
}

inline void VariableObj::setFloat(float f1, float f2, float f3)
{
  checkError( rtVariableSet3f( m_variable, f1, f2, f3 ) );
}

inline void VariableObj::setFloat(optix::float4 f)
{
  checkError( rtVariableSet4fv( m_variable, &f.x ) );
}

inline void VariableObj::setFloat(float f1, float f2, float f3, float f4)
{
  checkError( rtVariableSet4f( m_variable, f1, f2, f3, f4 ) );
}

inline void VariableObj::set1fv(const float* f)
{
  checkError( rtVariableSet1fv( m_variable, f ) );
}

inline void VariableObj::set2fv(const float* f)
{
  checkError( rtVariableSet2fv( m_variable, f ) );
}

inline void VariableObj::set3fv(const float* f)
{
  checkError( rtVariableSet3fv( m_variable, f ) );
}

inline void VariableObj::set4fv(const float* f)
{
  checkError( rtVariableSet4fv( m_variable, f ) );
}

inline void VariableObj::setInt(int i1)
{
  checkError( rtVariableSet1i( m_variable, i1 ) );
}

inline void VariableObj::setInt(optix::int2 i)
{
  checkError( rtVariableSet2iv( m_variable, &i.x ) );
}

inline void VariableObj::setInt(int i1, int i2)
{
  checkError( rtVariableSet2i( m_variable, i1, i2 ) );
}

inline void VariableObj::setInt(optix::int3 i)
{
  checkError( rtVariableSet3iv( m_variable, &i.x ) );
}

inline void VariableObj::setInt(int i1, int i2, int i3)
{
  checkError( rtVariableSet3i( m_variable, i1, i2, i3 ) );
}
```

```

inline void VariableObj::setInt(optix::int4 i)
{
  checkError( rtVariableSet4iv( m_variable, &i.x ) );
}

inline void VariableObj::setInt(int i1, int i2, int i3, int i4)
{
  checkError( rtVariableSet4i( m_variable, i1, i2, i3, i4 ) );
}

inline void VariableObj::set1iv( const int* i )
{
  checkError( rtVariableSet1iv( m_variable, i ) );
}

inline void VariableObj::set2iv( const int* i )
{
  checkError( rtVariableSet2iv( m_variable, i ) );
}

inline void VariableObj::set3iv( const int* i )
{
  checkError( rtVariableSet3iv( m_variable, i ) );
}

inline void VariableObj::set4iv( const int* i )
{
  checkError( rtVariableSet4iv( m_variable, i ) );
}

inline float VariableObj::getFloat() const
{
  float f;
  checkError( rtVariableGet1f( m_variable, &f ) );
  return f;
}

inline unsigned int VariableObj::getUint() const
{
  unsigned int i;
  checkError( rtVariableGet1ui( m_variable, &i ) );
  return i;
}

inline int VariableObj::getInt() const
{
  int i;
  checkError( rtVariableGet1i( m_variable, &i ) );
  return i;
}

inline void VariableObj::setBuffer(Buffer buffer)
{
  checkError( rtVariableSetObject( m_variable, buffer->get() ) );
}

inline void VariableObj::set(Buffer buffer)
{
  checkError( rtVariableSetObject( m_variable, buffer->get() ) );
}

inline void VariableObj::setUserData(RTsize size, const void* ptr)
{
  checkError( rtVariableSetUserData( m_variable, size, ptr ) );
}

inline void VariableObj::getUserData(RTsize size,       void* ptr) const
{
  checkError( rtVariableGetUserData( m_variable, size, ptr ) );
}

inline void VariableObj::setTextureSampler(TextureSampler texturesampler)
{
  checkError( rtVariableSetObject( m_variable, texturesampler->get() ) );
}

inline void VariableObj::set(TextureSampler texturesampler)
{
  checkError( rtVariableSetObject( m_variable, texturesampler->get() ) );
}

inline void VariableObj::set(GeometryGroup group)
{
  checkError( rtVariableSetObject( m_variable, group->get() ) );
}

inline void VariableObj::set(Group group)
{
  checkError( rtVariableSetObject( m_variable, group->get() ) );
}

inline void VariableObj::set(Program program)
{
  checkError( rtVariableSetObject( m_variable, program->get() ) );
}

inline void VariableObj::set(Selector sel)
{
  checkError( rtVariableSetObject( m_variable, sel->get() ) );
}

inline void VariableObj::set(Transform tran)
{
  checkError( rtVariableSetObject( m_variable, tran->get() ) );
}

inline Buffer VariableObj::getBuffer() const
{
  RTobject temp;
  checkError( rtVariableGetObject( m_variable, &temp ) );
  RTbuffer buffer = reinterpret_cast<RTbuffer>(temp);
  return Buffer::take(buffer);
}

inline std::string VariableObj::getName() const
{
  const char* name;
  checkError( rtVariableGetName( m_variable, &name ) );
  return std::string(name);
}

inline std::string VariableObj::getAnnotation() const
{
  const char* annotation;
  checkError( rtVariableGetAnnotation( m_variable, &annotation ) );
  return std::string(annotation);
}

inline RTobjecttype VariableObj::getType() const
{
  RTobjecttype type;
  checkError( rtVariableGetType( m_variable, &type ) );
  return type;
}

inline RTvariable VariableObj::get()
{
  return m_variable;
}

inline RTsize VariableObj::getSize() const
{
  RTsize size;
  checkError( rtVariableGetSize( m_variable, &size ) );
  return size;
}

```

```
  inline optix::TextureSampler VariableObj::getTextureSampler() const
  {
    RTobject temp;
    checkError( rtVariableGetObject( m_variable, &temp ) );
    RTtexturesampler sampler = reinterpret_cast<RTtexturesampler>(temp);
    return TextureSampler::take(sampler);
  }

  inline optix::Program VariableObj::getProgram() const
  {
    RTobject temp;
    checkError( rtVariableGetObject( m_variable, &temp ) );
    RTprogram program = reinterpret_cast<RTprogram>(temp);
    return Program::take(program);
  }

}

#endif /* __optixu_optixpp_namespace_h__ */


```

## 3.3 optixu.h File Reference

`#include <stddef.h> #include "../optix.h"`

### Defines

- #define RTU_INLINE inline
- #define RTU_CHECK_ERROR(func)
- #define RTU_GROUP_ADD_CHILD(_parent, _child, _index)
- #define RTU_SELECTOR_ADD_CHILD(_parent, _child, _index)

### Functions

- RTresult RTAPI rtuNameForType (RTobjecttype type, char ∗buffer, RTsize bufferSize)
- RTresult RTAPI rtuGetSizeForRTformat (RTformat format, size_t ∗size)
- RTresult RTAPI rtuCUDACompileString (const char ∗source, const char ∗∗preprocessorArguments, unsigned int numPreprocessorArguments, RTsize ∗resultSize, RTsize ∗errorSize)
- RTresult RTAPI rtuCUDACompileFile (const char ∗filename, const char ∗∗preprocessorArguments, unsigned int numPreprocessorArguments, RTsize ∗resultSize, RTsize ∗errorSize)
- RTresult RTAPI rtuCUDAGetCompileResult (char ∗result, char ∗error)
- RTresult rtuGroupAddChild (RTgroup group, RTobject child, unsigned int ∗index)
- RTresult rtuSelectorAddChild (RTselector selector, RTobject child, unsigned int ∗index)
- RTresult rtuGeometryGroupAddChild (RTgeometrygroup geometrygroup, RTgeometryinstance child, unsigned int ∗index)
- RTresult rtuTransformSetChild (RTtransform transform, RTobject child)
- RTresult rtuGroupRemoveChild (RTgroup group, RTobject child)
- RTresult rtuSelectorRemoveChild (RTselector selector, RTobject child)

- RTresult rtuGeometryGroupRemoveChild (RTgeometrygroup geometrygroup, R-Tgeometryinstance child)
- RTU_INLINE RTresult rtuGroupRemoveChildByIndex (RTgroup group, unsigned int index)
- RTU_INLINE RTresult rtuSelectorRemoveChildByIndex (RTselector selector, unsigned int index)
- RTU_INLINE RTresult rtuGeometryGroupRemoveChildByIndex (RTgeometrygroup geometrygroup, unsigned int index)
- RTU_INLINE RTresult rtuGroupGetChildIndex (RTgroup group, RTobject child, unsigned int ∗index)
- RTU_INLINE RTresult rtuSelectorGetChildIndex (RTselector selector, RTobject child, unsigned int ∗index)
- RTU_INLINE RTresult rtuGeometryGroupGetChildIndex (RTgeometrygroup geometrygroup, RTgeometryinstance child, unsigned int ∗index)
- RTresult RTAPI rtuCreateClusteredMesh (RTcontext context, unsigned int usePTX32InHost64, RTgeometry ∗mesh, unsigned int num_verts, const float ∗verts, unsigned int num_tris, const unsigned ∗indices, const unsigned ∗mat_indices)
- RTresult RTAPI rtuCreateClusteredMeshExt (RTcontext context, unsigned int usePTX32InHost64, RTgeometry ∗mesh, unsigned int num_verts, const float ∗verts, unsigned int num_tris, const unsigned ∗indices, const unsigned ∗mat_-indices, RTbuffer norms, const unsigned ∗norm_indices, RTbuffer tex_coords, const unsigned ∗tex_indices)

### 3.3.1 Define Documentation

#### 3.3.1.1 #define RTU_CHECK_ERROR( *func* )

**Value:**

```
do {                                              \
    RTresult code = func;                          \
    if( code != RT_SUCCESS )                       \
      return code;                                 \
  } while(0)
```

Definition at line 154 of file optixu.h.

#### 3.3.1.2 #define RTU_GROUP_ADD_CHILD( *_parent, _child, _index* )

**Value:**

```
unsigned int _count;                                                         \
  RTU_CHECK_ERROR( rtGroupGetChildCount( (_parent), &_count ) );               \
  RTU_CHECK_ERROR( rtGroupSetChildCount( (_parent), _count+1 ) );              \
  RTU_CHECK_ERROR( rtGroupSetChild( (_parent), _count, (_child) ) );           \
  if( _index ) *(_index) = _count;                                             \
  return RT_SUCCESS
```

Definition at line 161 of file optixu.h.

#### 3.3.1.3 #define RTU_INLINE inline

Definition at line 34 of file optixu.h.

#### 3.3.1.4 #define RTU_SELECTOR_ADD_CHILD( _parent, _child, _index )

**Value:**

```
unsigned int _count;                                                 \
  RTU_CHECK_ERROR( rtSelectorGetChildCount( (_parent), &_count ) );     \
  RTU_CHECK_ERROR( rtSelectorSetChildCount( (_parent), _count+1 ) );    \
  RTU_CHECK_ERROR( rtSelectorSetChild( (_parent), _count, (_child) ) ); \
  if( _index ) *(_index) = _count;                                      \
  return RT_SUCCESS
```

Definition at line 169 of file optixu.h.

### 3.3.2 Function Documentation

#### 3.3.2.1 RTresult RTAPI rtuCreateClusteredMesh ( RTcontext *context,* unsigned int *usePTX32InHost64,* RTgeometry ∗ *mesh,* unsigned int *num_verts,* const float ∗ *verts,* unsigned int *num_tris,* const unsigned ∗ *indices,* const unsigned ∗ *mat_indices* )

Create clustered triangle mesh for good memory coherence with paging on. Vertex, index and material buffers are created and attached to the mesh. Cluster's bounding box and intersection programs are attached to the mesh. The intersection program has the following attributes: rtDeclareVariable(float3, texcoord, attribute texcoord, ); It is always zero rtDeclareVariable(float3, geometric_normal, attribute geometric_normal, ); rtDeclareVariable(float3, shading_normal, attribute shading_normal, ); It is equal to geometric_normal

Created RTgeometry mesh expects there to be placed into a RTgeometryinstance where the mat_indices specified map into materials attached to the RTgeometryinstance

In the event of an error, please query the error string from the RTcontext.

**Parameters**

| | |
|---|---|
| *context* | Context |
| *usePTX32-InHost64* | Use 32bit PTX bounding box and intersection programs in 64bit application. Takes effect only with 64bit host. |
| *mesh* | Output geometry |
| *num_verts* | Vertex count |
| *verts* | Vertices (num_verts∗float∗3) [ v1_x, v1_y, v1_z, v2.x, ... ] |
| *num_tris* | Triangle count |
| *indices* | Vertex indices (num_tris∗unsigned∗3) [ tri1_index1, tr1_index2, ... ] |
| *mat_indices* | Indices of materials (num_tris∗unsigned) [ tri1_mat_index, tri2_mat_-index, ... ] |

#### 3.3.2.2 RTresult RTAPI rtuCreateClusteredMeshExt ( RTcontext *context,* unsigned int *usePTX32InHost64,* RTgeometry ∗ *mesh,* unsigned int *num_verts,* const float ∗ *verts,* unsigned int *num_tris,* const unsigned ∗ *indices,* const unsigned ∗ *mat_indices,* RTbuffer *norms,* const unsigned ∗ *norm_indices,* RTbuffer *tex_coords,* const unsigned ∗ *tex_indices* )

Create clustered triangle mesh for good memory coherence with paging on. Buffers for vertices, indices, normals, indices of normals, texture coordinates, indices of texture coordinates and materials are created and attached to the mesh. Cluster's bounding box and intersection programs are attached to the mesh. The intersection program has the following attributes: rtDeclareVariable(float3, texcoord, attribute texcoord, ); rtDeclareVariable(float3, geometric_normal, attribute geometric_normal, ); rtDeclare-Variable(float3, shading_normal, attribute shading_normal, );

Created RTgeometry mesh expects there to be placed into a RTgeometryinstance where the mat_indices specified map into materials attached to the RTgeometryinstance

Vertex, normal and texture coordinate buffers can be shared between many geometry objects

In the event of an error, please query the error string from the RTcontext.

**Parameters**

| | |
|---|---|
| *context* | Context |
| *usePTX32-InHost64* | Use 32bit PTX bounding box and intersection programs in 64bit application. Takes effect only with 64bit host. |
| *mesh* | Output geometry |
| *num_verts* | Vertex count |
| *verts* | Vertices (num_verts∗float∗3) [ v1_x, v1_y, v1_z, v2.x, ... ] |
| *num_tris* | Triangle count |
| *indices* | Vertex indices (num_tris∗unsigned∗3) [ tri1_index1, tr1_index2, ... ] |
| *mat_indices* | Indices of materials (num_tris∗unsigned) [ tri1_mat_index, tri2_mat_-index, ... ] |
| *norms* | Normals (num_norms∗float∗3) [ v1_x, v1_y, v1_z, v2.x, ... ] |
| *norm_-indices* | Indices of vertex normals (num_tris∗unsigned∗3) [ tri1_norm_index1, tri1_norm_index2 ... ] |
| *tex_coords* | Texture uv coords (num_tex_coords∗float∗2) [ t1_u, t1_v, t2_u ... ] |
| *tex_indices* | Indices of texture uv (num_tris∗unsigned∗3) [ tri1_tex_index1, tri1_tex-_index2 ... ] |

#### 3.3.2.3 RTresult RTAPI rtuCUDACompileFile ( const char ∗ *filename,* const char ∗∗ *preprocessorArguments,* unsigned int *numPreprocessorArguments,* RTsize ∗ *resultSize,* RTsize ∗ *errorSize* )

#### 3.3.2.4 RTresult RTAPI rtuCUDACompileString ( const char ∗ *source,* const char ∗∗ *preprocessorArguments,* unsigned int *numPreprocessorArguments,* RTsize ∗ *resultSize,* RTsize ∗ *errorSize* )

#### 3.3.2.5 RTresult RTAPI rtuCUDAGetCompileResult ( char ∗ *result,* char ∗ *error* )

#### 3.3.2.6 RTU_INLINE RTresult rtuGeometryGroupAddChild ( RTgeometrygroup *geometrygroup,* RTgeometryinstance *child,* unsigned int ∗ *index* )

Definition at line 273 of file optixu.h.

#### 3.3.2.7 RTU_INLINE RTresult rtuGeometryGroupGetChildIndex ( RTgeometrygroup *geometrygroup,* RTgeometryinstance *child,* unsigned int ∗ *index* )

Definition at line 366 of file optixu.h.

#### 3.3.2.8 RTU_INLINE RTresult rtuGeometryGroupRemoveChild ( RTgeometrygroup *geometrygroup,* RTgeometryinstance *child* )

Definition at line 299 of file optixu.h.

#### 3.3.2.9 RTU_INLINE RTresult rtuGeometryGroupRemoveChildByIndex ( RTgeometrygroup *geometrygroup,* unsigned int *index* )

Definition at line 329 of file optixu.h.

#### 3.3.2.10 RTresult RTAPI rtuGetSizeForRTformat ( RTformat *format,* size_t ∗ *size* )

#### 3.3.2.11 RTU_INLINE RTresult rtuGroupAddChild ( RTgroup *group,* RTobject *child,* unsigned int ∗ *index* )

Definition at line 180 of file optixu.h.

#### 3.3.2.12 RTU_INLINE RTresult rtuGroupGetChildIndex ( RTgroup *group,* RTobject *child,* unsigned int ∗ *index* )

Definition at line 340 of file optixu.h.

#### 3.3.2.13 RTU_INLINE RTresult rtuGroupRemoveChild ( RTgroup *group,* RTobject *child* )

Definition at line 283 of file optixu.h.

#### 3.3.2.14 RTU_INLINE RTresult rtuGroupRemoveChildByIndex ( RTgroup *group,* unsigned int *index* )

Definition at line 307 of file optixu.h.

#### 3.3.2.15 RTresult RTAPI rtuNameForType ( RTobjecttype *type,* char ∗ *buffer,* RTsize *bufferSize* )

#### 3.3.2.16 RTU_INLINE RTresult rtuSelectorAddChild ( RTselector *selector,* RTobject *child,* unsigned int ∗ *index* )

Definition at line 185 of file optixu.h.

#### 3.3.2.17 RTU_INLINE RTresult rtuSelectorGetChildIndex ( RTselector *selector,* RTobject *child,* unsigned int ∗ *index* )

Definition at line 353 of file optixu.h.

#### 3.3.2.18 RTU_INLINE RTresult rtuSelectorRemoveChild ( RTselector *selector,* RTobject *child* )

Definition at line 291 of file optixu.h.

#### 3.3.2.19 RTU_INLINE RTresult rtuSelectorRemoveChildByIndex ( RTselector *selector,* unsigned int *index* )

Definition at line 318 of file optixu.h.

#### 3.3.2.20 RTU_INLINE RTresult rtuTransformSetChild ( RTtransform *transform,* RTobject *child* )

Definition at line 239 of file optixu.h.

## 3.4 optixu.h

```

/*
 * Copyright (c) 2008 - 2009 NVIDIA Corporation.  All rights reserved.
 *
 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
 * rights in and to this software, related documentation and any modifications thereto.
 * Any use, reproduction, disclosure or distribution of this software and related
 * documentation without an express license agreement from NVIDIA Corporation is strictly
 * prohibited.
 *
 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGES
 */

#ifndef __optix_optixu_h__
#define __optix_optixu_h__

#include <stddef.h>
#include "../optix.h"

#ifdef __cplusplus
#  define RTU_INLINE inline
#else
```

```
#  ifdef _MSC_VER
#    define RTU_INLINE __inline
#  else
#    define RTU_INLINE inline
#  endif
#endif

#ifdef __cplusplus
extern "C" {
#endif

 /*
  * Get the name string of a given type.
  */
  RTresult RTAPI rtuNameForType( RTobjecttype type, char* buffer, RTsize
bufferSize );

 /*
  * Return the size of a given RTformat.  RT_FORMAT_USER and RT_FORMAT_UNKNOWN
 return 0.
  * Returns RT_ERROR_INVALID_VALUE if the format isn't recognized, RT_SUCCESS
 otherwise.
  */
  RTresult RTAPI rtuGetSizeForRTformat( RTformat format, size_t* size);

 /*
  * Compile a cuda source string.
  * ARGS:
  *
  * source                          source code string
  * preprocessorArguments           list of preprocessor arguments
  * numPreprocessorArguments        number of preprocessor arguments
  * resultSize                      [out] size required to hold compiled result
 string
  * errorSize                       [out] size required to hold error string
  */
  RTresult RTAPI rtuCUDACompileString( const char* source, const char**
preprocessorArguments, unsigned int numPreprocessorArguments, RTsize* resultSize,
RTsize* errorSize );

 /*
  * Compile a cuda source file.
  * ARGS:
  *
  * filename                        source code file name
  * preprocessorArguments           list of preprocessor arguments
  * numPreprocessorArguments        number of preprocessor arguments
  * resultSize                      [out] size required to hold compiled result
 string
  * errorSize                       [out] size required to hold error string
  */
  RTresult RTAPI rtuCUDACompileFile( const char* filename, const char**
preprocessorArguments, unsigned int numPreprocessorArguments, RTsize* resultSize,
RTsize* errorSize );

 /*
  * Get the result of the most recent call to one of the above compile
 functions.
  * The 'result' and 'error' parameters must point to memory large enough to
 hold
  * the respective strings, as returned by the compile function.
  * ARGS:
  *
  * result                        compiled result string
  * error                         error string
  */
  RTresult RTAPI rtuCUDAGetCompileResult( char* result, char* error );

#ifdef __cplusplus
} /* extern "C" */
#endif

 /*
  * Add an entry to the end of the child array.
```

```
   * Fills 'index' with the index of the added child, if the pointer is non-NULL.
   */
#ifndef __cplusplus
  RTresult rtuGroupAddChild        ( RTgroup group, RTobject child, unsigned int* index );
  RTresult rtuSelectorAddChild     ( RTselector selector, RTobject child, unsigned int* index );
#else
  RTresult rtuGroupAddChild        ( RTgroup group, RTgroup         child, unsigned int* index );
  RTresult rtuGroupAddChild        ( RTgroup group, RTselector      child, unsigned int* index );
  RTresult rtuGroupAddChild        ( RTgroup group, RTtransform     child, unsigned int* index );
  RTresult rtuGroupAddChild        ( RTgroup group, RTgeometrygroup child, unsigned int* index );
  RTresult rtuSelectorAddChild     ( RTselector selector, RTgroup         child, unsigned int* index );
  RTresult rtuSelectorAddChild     ( RTselector selector, RTselector      child, unsigned int* index );
  RTresult rtuSelectorAddChild     ( RTselector selector, RTtransform     child, unsigned int* index );
  RTresult rtuSelectorAddChild     ( RTselector selector, RTgeometrygroup child, unsigned int* index );
#endif
  RTresult rtuGeometryGroupAddChild( RTgeometrygroup geometrygroup, RTgeometryinstance child, unsigned int* index );

  /*
   * Wrap rtTransformSetChild in order to provide a type-safe version for C++.
   */
#ifndef __cplusplus
  RTresult rtuTransformSetChild    ( RTtransform transform, RTobject child );
#else
  RTresult rtuTransformSetChild    ( RTtransform transform, RTgroup child );
  RTresult rtuTransformSetChild    ( RTtransform transform, RTselector child );
  RTresult rtuTransformSetChild    ( RTtransform transform, RTtransform child );
  RTresult rtuTransformSetChild    ( RTtransform transform, RTgeometrygroup child );
#endif

  /*
   * Find the given child using a linear search in the child array and remove
   * it. If it's not the last entry in the child array, the last entry in the
   * array will replace the deleted entry, in order to shrink the array size by one.
   */
  RTresult rtuGroupRemoveChild        ( RTgroup group, RTobject child );
  RTresult rtuSelectorRemoveChild     ( RTselector selector, RTobject child );
  RTresult rtuGeometryGroupRemoveChild( RTgeometrygroup geometrygroup, RTgeometryinstance child );

  /*
   * Remove the child at the given index in the child array. If it's not the last
   * entry in the child array, the last entry in the array will replace the deleted
   * entry, in order to shrink the array size by one.
   */
  RTU_INLINE RTresult rtuGroupRemoveChildByIndex        ( RTgroup group, unsigned int index );
  RTU_INLINE RTresult rtuSelectorRemoveChildByIndex     ( RTselector selector, unsigned int index );
  RTU_INLINE RTresult rtuGeometryGroupRemoveChildByIndex( RTgeometrygroup geometrygroup, unsigned int index );

  /*
   * Use a linear search to find the child in the child array, and return its index.
   * Returns RT_SUCCESS if the child was found, RT_INVALID_VALUE otherwise.
   */
  RTU_INLINE RTresult rtuGroupGetChildIndex        ( RTgroup group, RTobject
 child, unsigned int* index );
  RTU_INLINE RTresult rtuSelectorGetChildIndex     ( RTselector selector,
 RTobject child, unsigned int* index );
  RTU_INLINE RTresult rtuGeometryGroupGetChildIndex( RTgeometrygroup
 geometrygroup, RTgeometryinstance child, unsigned int* index );


  /*
   * The following implements the child management helpers declared above.
   */

 #define RTU_CHECK_ERROR( func )                  \
   do {                                           \
     RTresult code = func;                        \
     if( code != RT_SUCCESS )                     \
       return code;                               \
   } while(0)

 #define RTU_GROUP_ADD_CHILD( _parent, _child, _index )                   \
   unsigned int _count;                                                   \
   RTU_CHECK_ERROR( rtGroupGetChildCount( (_parent), &_count ) );         \
   RTU_CHECK_ERROR( rtGroupSetChildCount( (_parent), _count+1 ) );        \
   RTU_CHECK_ERROR( rtGroupSetChild( (_parent), _count, (_child) ) );     \
   if( _index ) *(_index) = _count;                                       \
   return RT_SUCCESS

 #define RTU_SELECTOR_ADD_CHILD( _parent, _child, _index )                \
   unsigned int _count;                                                   \
   RTU_CHECK_ERROR( rtSelectorGetChildCount( (_parent), &_count ) );      \
   RTU_CHECK_ERROR( rtSelectorSetChildCount( (_parent), _count+1 ) );     \
   RTU_CHECK_ERROR( rtSelectorSetChild( (_parent), _count, (_child) ) ); \
   if( _index ) *(_index) = _count;                                       \
   return RT_SUCCESS


 #ifndef __cplusplus

  RTU_INLINE RTresult rtuGroupAddChild( RTgroup group, RTobject child, unsigned
 int* index )
  {
    RTU_GROUP_ADD_CHILD( group, child, index );
  }

  RTU_INLINE RTresult rtuSelectorAddChild( RTselector selector, RTobject child,
 unsigned int* index )
  {
    RTU_SELECTOR_ADD_CHILD( selector, child, index );
  }

 #else /* __cplusplus */

  RTU_INLINE RTresult rtuGroupAddChild( RTgroup group, RTgroup child, unsigned
 int* index )
  {
    RTU_GROUP_ADD_CHILD( group, child, index );
  }

  RTU_INLINE RTresult rtuGroupAddChild( RTgroup group, RTselector child,
 unsigned int* index )
  {
    RTU_GROUP_ADD_CHILD( group, child, index );
  }

  RTU_INLINE RTresult rtuGroupAddChild( RTgroup group, RTtransform child,
 unsigned int* index )
  {
    RTU_GROUP_ADD_CHILD( group, child, index );
  }

  RTU_INLINE RTresult rtuGroupAddChild( RTgroup group, RTgeometrygroup child,
 unsigned int* index )
```

```
 {
   RTU_GROUP_ADD_CHILD( group, child, index );
 }

 RTU_INLINE RTresult rtuSelectorAddChild( RTselector selector, RTgroup child, unsigned int* index )
 {
   RTU_SELECTOR_ADD_CHILD( selector, child, index );
 }

 RTU_INLINE RTresult rtuSelectorAddChild( RTselector selector, RTselector child, unsigned int* index )
 {
   RTU_SELECTOR_ADD_CHILD( selector, child, index );
 }

 RTU_INLINE RTresult rtuSelectorAddChild( RTselector selector, RTtransform child, unsigned int* index )
 {
   RTU_SELECTOR_ADD_CHILD( selector, child, index );
 }

 RTU_INLINE RTresult rtuSelectorAddChild( RTselector selector, RTgeometrygroup child, unsigned int* index )
 {
   RTU_SELECTOR_ADD_CHILD( selector, child, index );
 }

#endif /* __cplusplus */

#undef RTU_GROUP_ADD_CHILD
#undef RTU_SELECTOR_ADD_CHILD

#ifndef __cplusplus

 RTU_INLINE RTresult rtuTransformSetChild( RTtransform transform, RTobject child )
 {
   RTU_CHECK_ERROR( rtTransformSetChild( transform, child ) );
   return RT_SUCCESS;
 }

#else /* __cplusplus */

 RTU_INLINE RTresult rtuTransformSetChild( RTtransform transform, RTgroup child )
 {
   RTU_CHECK_ERROR( rtTransformSetChild( transform, child ) );
   return RT_SUCCESS;
 }

 RTU_INLINE RTresult rtuTransformSetChild( RTtransform transform, RTselector child )
 {
   RTU_CHECK_ERROR( rtTransformSetChild( transform, child ) );
   return RT_SUCCESS;
 }

 RTU_INLINE RTresult rtuTransformSetChild( RTtransform transform, RTtransform child )
 {
   RTU_CHECK_ERROR( rtTransformSetChild( transform, child ) );
   return RT_SUCCESS;
 }

 RTU_INLINE RTresult rtuTransformSetChild( RTtransform transform, RTgeometrygroup child )
 {
   RTU_CHECK_ERROR( rtTransformSetChild( transform, child ) );
   return RT_SUCCESS;
 }

#endif /* __cplusplus */

```

```
RTU_INLINE RTresult rtuGeometryGroupAddChild( RTgeometrygroup geometrygroup, RTgeometryinstance child, unsigned int* index )
{
  unsigned int count;
  RTU_CHECK_ERROR( rtGeometryGroupGetChildCount( geometrygroup, &count ) );
  RTU_CHECK_ERROR( rtGeometryGroupSetChildCount( geometrygroup, count+1 ) );
  RTU_CHECK_ERROR( rtGeometryGroupSetChild( geometrygroup, count, child ) );
  if( index ) *index = count;
  return RT_SUCCESS;
}

RTU_INLINE RTresult rtuGroupRemoveChild( RTgroup group, RTobject child )
{
  unsigned int index;
  RTU_CHECK_ERROR( rtuGroupGetChildIndex( group, child, &index ) );
  RTU_CHECK_ERROR( rtuGroupRemoveChildByIndex( group, index ) );
  return RT_SUCCESS;
}

RTU_INLINE RTresult rtuSelectorRemoveChild( RTselector selector, RTobject child )
{
  unsigned int index;
  RTU_CHECK_ERROR( rtuSelectorGetChildIndex( selector, child, &index ) );
  RTU_CHECK_ERROR( rtuSelectorRemoveChildByIndex( selector, index ) );
  return RT_SUCCESS;
}

RTU_INLINE RTresult rtuGeometryGroupRemoveChild( RTgeometrygroup geometrygroup, RTgeometryinstance child )
{
  unsigned int index;
  RTU_CHECK_ERROR( rtuGeometryGroupGetChildIndex( geometrygroup, child, &index ) );
  RTU_CHECK_ERROR( rtuGeometryGroupRemoveChildByIndex( geometrygroup, index ) );
  return RT_SUCCESS;
}

RTU_INLINE RTresult rtuGroupRemoveChildByIndex( RTgroup group, unsigned int index )
{
  unsigned int count;
  RTobject temp;
  RTU_CHECK_ERROR( rtGroupGetChildCount( group, &count ) );
  RTU_CHECK_ERROR( rtGroupGetChild( group, count-1, &temp ) );
  RTU_CHECK_ERROR( rtGroupSetChild( group, index, temp ) );
  RTU_CHECK_ERROR( rtGroupSetChildCount( group, count-1 ) );
  return RT_SUCCESS;
}

RTU_INLINE RTresult rtuSelectorRemoveChildByIndex( RTselector selector, unsigned int index )
{
  unsigned int count;
  RTobject temp;
  RTU_CHECK_ERROR( rtSelectorGetChildCount( selector, &count ) );
  RTU_CHECK_ERROR( rtSelectorGetChild( selector, count-1, &temp ) );
  RTU_CHECK_ERROR( rtSelectorSetChild( selector, index, temp ) );
  RTU_CHECK_ERROR( rtSelectorSetChildCount( selector, count-1 ) );
  return RT_SUCCESS;
}

RTU_INLINE RTresult rtuGeometryGroupRemoveChildByIndex( RTgeometrygroup geometrygroup, unsigned int index )
{
  unsigned int count;
  RTgeometryinstance temp;
  RTU_CHECK_ERROR( rtGeometryGroupGetChildCount( geometrygroup, &count ) );
  RTU_CHECK_ERROR( rtGeometryGroupGetChild( geometrygroup, count-1, &temp ) );
  RTU_CHECK_ERROR( rtGeometryGroupSetChild( geometrygroup, index, temp ) );
  RTU_CHECK_ERROR( rtGeometryGroupSetChildCount( geometrygroup, count-1 ) );
  return RT_SUCCESS;
}
```

```

 RTU_INLINE RTresult rtuGroupGetChildIndex(RTgroup group, RTobject child, unsigned int* index)
 {
   unsigned int count;
   RTobject temp;
   RTU_CHECK_ERROR( rtGroupGetChildCount( group, &count ) );
   for( *index=0; *index<count; (*index)++ ) {
     RTU_CHECK_ERROR( rtGroupGetChild( group, *index, &temp ) );
     if( child==temp ) return RT_SUCCESS;
   }
   *index = ~0u;
   return RT_ERROR_INVALID_VALUE;
 }

 RTU_INLINE RTresult rtuSelectorGetChildIndex( RTselector selector, RTobject child, unsigned int* index )
 {
   unsigned int count;
   RTobject temp;
   RTU_CHECK_ERROR( rtSelectorGetChildCount( selector, &count ) );
   for( *index=0; *index<count; (*index)++ ) {
     RTU_CHECK_ERROR( rtSelectorGetChild( selector, *index, &temp ) );
     if( child==temp ) return RT_SUCCESS;
   }
   *index = ~0u;
   return RT_ERROR_INVALID_VALUE;
 }

 RTU_INLINE RTresult rtuGeometryGroupGetChildIndex( RTgeometrygroup geometrygroup, RTgeometryinstance child, unsigned int* index )
 {
   unsigned int count;
   RTgeometryinstance temp;
   RTU_CHECK_ERROR( rtGeometryGroupGetChildCount( geometrygroup, &count ) );
   for( *index=0; *index<count; (*index)++ ) {
     RTU_CHECK_ERROR( rtGeometryGroupGetChild( geometrygroup, *index, &temp ) );
     if( child==temp ) return RT_SUCCESS;
   }
   *index = ~0u;
   return RT_ERROR_INVALID_VALUE;
 }


#ifdef __cplusplus
extern "C" {
#endif

  RTresult RTAPI rtuCreateClusteredMesh( RTcontext       context,
                                         unsigned int    usePTX32InHost64,
                                         RTgeometry*     mesh,
                                         unsigned int    num_verts,
                                         const float*    verts,
                                         unsigned int    num_tris,
                                         const unsigned* indices,
                                         const unsigned* mat_indices);



  RTresult RTAPI rtuCreateClusteredMeshExt( RTcontext       context,
                                            unsigned int    usePTX32InHost64,
                                            RTgeometry*     mesh,
                                            unsigned int    num_verts,
                                            const float*    verts,
                                            unsigned int    num_tris,
                                            const unsigned* indices,
                                            const unsigned* mat_indices,
                                            RTbuffer        norms,
                                            const unsigned* norm_indices,
                                            RTbuffer        tex_coords,
                                            const unsigned* tex_indices );

#ifdef __cplusplus
} /* extern "C" */
#endif


#undef RTU_CHECK_ERROR
#undef RTU_INLINE

#endif /* __optix_optixu_h__ */
```

## 3.5 optixu_traversal.h File Reference

```
#include "../optix.h"
```

### Classes

- struct RTUtraversalresult

    *Structure encapsulating the result of a single ray query.*

### Typedefs

- typedef struct RTUtraversal_api ∗ RTUtraversal

### Enumerations

- enum RTUquerytype { RTU_QUERY_TYPE_ANY_HIT = 0, RTU_QUERY_TYPE_CLOSEST_HIT, RTU_QUERY_TYPE_COUNT }
- enum RTUrayformat { RTU_RAYFORMAT_ORIGIN_DIRECTION_TMIN_TMAX_INTERLEAVED = 0, RTU_RAYFORMAT_ORIGIN_DIRECTION_INTERLEAVED, RTU_RAYFORMAT_COUNT }
- enum RTUtriformat { RTU_TRIFORMAT_MESH = 0, RTU_TRIFORMAT_TRIANGLE_SOUP, RTU_TRIFORMAT_COUNT }
- enum RTUinitoptions { RTU_INITOPTION_NONE = 0, RTU_INITOPTION_GPU_ONLY = 1 << 0, RTU_INITOPTION_CPU_ONLY = 1 << 1, RTU_INITOPTION_CULL_BACKFACE = 1 << 2 }
- enum RTUoutput { RTU_OUTPUT_NONE = 0, RTU_OUTPUT_NORMAL = 1 << 0, RTU_OUTPUT_BARYCENTRIC = 1 << 1, RTU_OUTPUT_BACKFACING = 1 << 2 }
- enum RTUoption { RTU_OPTION_INT_NUM_THREADS = 0 }

### Functions

- RTresult RTAPI rtuTraversalCreate (RTUtraversal ∗traversal, RTUquerytype query_type, RTUrayformat ray_format, RTUtriformat tri_format, unsigned int outputs, unsigned int options, RTcontext context)
- RTresult RTAPI rtuTraversalGetErrorString (RTUtraversal traversal, RTresult code, const char ∗∗return_string)
- RTresult RTAPI rtuTraversalSetOption (RTUtraversal traversal, RTUoption option, void ∗value)

- RTresult RTAPI rtuTraversalSetMesh (RTUtraversal traversal, unsigned int num-_verts, const float ∗verts, unsigned int num_tris, const unsigned ∗indices)
- RTresult RTAPI rtuTraversalSetTriangles (RTUtraversal traversal, unsigned int num_tris, const float ∗tris)
- RTresult RTAPI rtuTraversalSetAccelData (RTUtraversal traversal, const void ∗data, RTsize data_size)
- RTresult RTAPI rtuTraversalGetAccelDataSize (RTUtraversal traversal, RTsize ∗data_size)
- RTresult RTAPI rtuTraversalGetAccelData (RTUtraversal traversal, void ∗data)
- RTresult RTAPI rtuTraversalMapRays (RTUtraversal traversal, unsigned int num-_rays, float ∗∗rays)
- RTresult RTAPI rtuTraversalUnmapRays (RTUtraversal traversal)
- RTresult RTAPI rtuTraversalPreprocess (RTUtraversal traversal)
- RTresult RTAPI rtuTraversalTraverse (RTUtraversal traversal)
- RTresult RTAPI rtuTraversalMapResults (RTUtraversal traversal, RTUtraversalresult ∗∗results)
- RTresult RTAPI rtuTraversalUnmapResults (RTUtraversal traversal)
- RTresult RTAPI rtuTraversalMapOutput (RTUtraversal traversal, RTUoutput which, void ∗∗output)
- RTresult RTAPI rtuTraversalUnmapOutput (RTUtraversal traversal, RTUoutput which)
- RTresult RTAPI rtuTraversalDestroy (RTUtraversal traversal)

### 3.5.1 Detailed Description

A simple API for performing raytracing queries using OptiX or the CPU.

Definition in file optixu_traversal.h.

### 3.5.2 Typedef Documentation

#### 3.5.2.1 typedef struct RTUtraversal_api∗ **RTUtraversal**

Opaque type. Note that the ∗_api types should never be used directly. Only the typedef target names will be guaranteed to remain unchanged.

Definition at line 116 of file optixu_traversal.h.

### 3.5.3 Enumeration Type Documentation

#### 3.5.3.1 enum **RTUinitoptions**

Initialization options (static across life of traversal object).

The rtuTraverse API supports both running on the CPU and GPU. When RTU_INIT-OPTION_NONE is specified GPU context creation is attempted. If that fails (such as when there isn't an NVIDIA GPU part present, the CPU code path is automatically chosen. Specifying RTU_INITOPTION_GPU_ONLY or RTU_INITOPTION_CPU_ONLY will only use the GPU or CPU modes without automatic transitions from one to the other.

RTU_INITOPTION_CULL_BACKFACE will enable back face culling during intersection.

**Enumerator:**

***RTU_INITOPTION_NONE***
***RTU_INITOPTION_GPU_ONLY***
***RTU_INITOPTION_CPU_ONLY***
***RTU_INITOPTION_CULL_BACKFACE***

Definition at line 89 of file optixu_traversal.h.

#### 3.5.3.2 enum RTUoption

Runtime options (can be set multiple times for a given traversal object).

**Enumerator:**

***RTU_OPTION_INT_NUM_THREADS***

Definition at line 107 of file optixu_traversal.h.

#### 3.5.3.3 enum RTUoutput

**Enumerator:**

***RTU_OUTPUT_NONE***
***RTU_OUTPUT_NORMAL***
***RTU_OUTPUT_BARYCENTRIC***
***RTU_OUTPUT_BACKFACING***

Definition at line 96 of file optixu_traversal.h.

#### 3.5.3.4 enum RTUquerytype

The type of ray query to be performed.

See OptiX Programming Guide for explanation of any vs. closest hit queries. Note that in the case of RTU_QUERY_TYPE_ANY_HIT, the prim_id and t intersection values in RTUtraversalresult will correspond to the first successful intersection. These values may not be indicative of the closest intersection, only that there was at least one.

**Enumerator:**

***RTU_QUERY_TYPE_ANY_HIT*** Perform any hit calculation
***RTU_QUERY_TYPE_CLOSEST_HIT*** Perform closest hit calculation
***RTU_QUERY_TYPE_COUNT***

Definition at line 49 of file optixu_traversal.h.

#### 3.5.3.5 enum RTUrayformat

The input format of the ray vector.

**Enumerator:**

***RTU_RAYFORMAT_ORIGIN_DIRECTION_TMIN_TMAX_INTERLEAVED***

***RTU_RAYFORMAT_ORIGIN_DIRECTION_INTERLEAVED***

***RTU_RAYFORMAT_COUNT***

Definition at line 58 of file optixu_traversal.h.

#### 3.5.3.6 enum RTUtriformat

The input format of the triangles.

TRIANGLE_SOUP implies future use of rtuTraversalSetTriangles while MESH implies use of rtuTraversalSetMesh.

**Enumerator:**

***RTU_TRIFORMAT_MESH***

***RTU_TRIFORMAT_TRIANGLE_SOUP***

***RTU_TRIFORMAT_COUNT***

Definition at line 70 of file optixu_traversal.h.

### 3.5.4 Function Documentation

#### 3.5.4.1 RTresult RTAPI rtuTraversalCreate ( RTUtraversal ∗ *traversal,* RTUquerytype *query_type,* RTUrayformat *ray_format,* RTUtriformat *tri_format,* unsigned int *outputs,* unsigned int *options,* RTcontext *context* )

Create a traversal state and associate a context with it. If context is a null pointer a new context will be created internally. The context should also not be used for any other launch commands from the OptiX host API, nor attached to multiple RTUtraversal objects at one time.

**Parameters**

| | | |
|---|---|---|
| out | *traversal* | Return pointer for traverse state handle |
| | *query_type* | Ray query type |
| | *ray_format* | Ray format |
| | *tri_format* | Triangle format |
| | *outputs* | OR'ed mask of requested RTUoutputs |
| | *options* | Bit vector of or'ed RTUinitoptions. |
| | *context* | RTcontext used for internal object creation |

#### 3.5.4.2 RTresult RTAPI **rtuTraversalDestroy** ( **RTUtraversal** *traversal* )

Clean up any internal memory associated with rtuTraversal operations. Includes destruction of result buffers returned via rtuTraversalGetResults. Invalidates traversal object.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |

#### 3.5.4.3 RTresult RTAPI **rtuTraversalGetAccelData** ( **RTUtraversal** *traversal,* void ∗ *data* )

Retrieve acceleration data for current geometry. Will force acceleration build if necessary. The data parameter should be preallocated and its length should match return value of rtuTraversalGetAccelDataSize.

**Parameters**

| | | |
|---|---|---|
| | *traversal* | Traversal state handle |
| `out` | *data* | Acceleration data |

#### 3.5.4.4 RTresult RTAPI **rtuTraversalGetAccelDataSize** ( **RTUtraversal** *traversal,* RTsize ∗ *data_size* )

Retrieve acceleration data size for current geometry. Will force acceleration build if necessary.

**Parameters**

| | | |
|---|---|---|
| | *traversal* | Traversal state handle |
| `out` | *data_size* | Size of acceleration data |

#### 3.5.4.5 RTresult RTAPI **rtuTraversalGetErrorString** ( **RTUtraversal** *traversal,* RTresult *code,* const char ∗∗ *return_string* )

Returns the string associated with the error code and any additional information from the last error. If traversal is non-NULL return_string only remains valid while traversal is live.

**Parameters**

| | | |
|---|---|---|
| | *traversal* | Traversal state handle. Can be NULL. |
| | *code* | Error code from last error |
| `out` | *return_string* | Pointer to string with error message in it. |

#### 3.5.4.6 RTresult RTAPI **rtuTraversalMapOutput** ( **RTUtraversal** *traversal,* **RTUoutput** *which,* void ∗∗ *output* )

Retrieve user-specified output from last rtuTraversal call. Output can be copied from the pointer returned by rtuTraversalMapOutput and will have length 'num_rays' from as prescribed from the previous call to rtuTraversalSetRays. For each RTUoutput, a single rtuTraversalMapOutput pointers can be outstanding. rtuTraversalUnmapOutput should be called when finished reading the output.

If requested output type was not turned on with a previous call to rtuTraverseSetOutputs an error will be returned. See RTUoutput enum for description of output data formats for various outputs.

**Parameters**

| | | |
|---|---|---|
| | *traversal* | Traversal state handle |
| | *which* | Output type to be specified |
| `out` | *output* | Pointer to output from last traverse |

#### 3.5.4.7 RTresult RTAPI **rtuTraversalMapRays** ( **RTUtraversal** *traversal,* unsigned int *num_rays,* float ** *rays* )

Specify set of rays to be cast upon next call to rtuTraversalTraverse. rtuTraversalMapRays obtains a pointer which can be used to copy the ray data into. Rays should be packed in the format described in rtuTraversalCreate call. When copying is completed rtuTraversalUnmapRays should be called. Note that this call invalidates any existing results buffers until rtuTraversalTraverse is called again.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |
| *num_rays* | Number of rays to be traced |
| *rays* | Pointer to ray data |

#### 3.5.4.8 RTresult RTAPI **rtuTraversalMapResults** ( **RTUtraversal** *traversal,* **RTUtraversalresult** ** *results* )

Retrieve results of last rtuTraversal call. Results can be copied from the pointer returned by rtuTraversalMapResults and will have length 'num_rays' as prescribed from the previous call to rtuTraversalMapRays. rtuTraversalUnmapResults should be called when finished reading the results. Returned primitive ID of -1 indicates a ray miss.

**Parameters**

| | | |
|---|---|---|
| | *traversal* | Traversal state handle |
| `out` | *results* | Pointer to results of last traverse |

#### 3.5.4.9 RTresult RTAPI **rtuTraversalPreprocess** ( **RTUtraversal** *traversal* )

Perform any necessary preprocessing (eg, acceleration structure building, optix context compilation). It is not necessary to call this function as rtuTraversalTraverse will call this internally as necessary.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |

#### 3.5.4.10 RTresult RTAPI rtuTraversalSetAccelData ( RTUtraversal *traversal,* const void ∗ *data,* RTsize *data_size* )

Specify acceleration data for current geometry. Input acceleration data should be result of rtuTraversalGetAccelData or rtAccelerationGetData call.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |
| *data* | Acceleration data |
| *data_size* | Size of acceleration data |

#### 3.5.4.11 RTresult RTAPI rtuTraversalSetMesh ( RTUtraversal *traversal,* unsigned int *num_verts,* const float ∗ *verts,* unsigned int *num_tris,* const unsigned ∗ *indices* )

Specify triangle mesh to be intersected by the next call to rtuTraversalLaunch. Only one geometry set may be active at a time. Subsequent calls to rtuTraversalSetTriangles or rtuTraversalSetMesh will override any previously specified geometry. No internal copies of the mesh data are made. The user should ensure that the mesh data remains valid until after rtuTraversalTraverse has been called. Counter-clockwise winding is assumed for normal and backfacing computations.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |
| *num_verts* | Vertex count |
| *verts* | Vertices [ v1_x, v1_y, v1_z, v2.x, ... ] |
| *num_tris* | Triangle count |
| *indices* | Indices [ tri1_index1, tr1_index2, ... ] |

#### 3.5.4.12 RTresult RTAPI rtuTraversalSetOption ( RTUtraversal *traversal,* RTUoption *option,* void ∗ *value* )

Set a runtime option. Unlike initialization options, these options may be set more than once for a given RTUtraversal instance.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |
| *option* | The option to be set |
| *value* | Value of the option |

#### 3.5.4.13 RTresult RTAPI rtuTraversalSetTriangles ( RTUtraversal *traversal,* unsigned int *num_tris,* const float ∗ *tris* )

Specify triangle soup to be intersected by the next call to rtuTraversalLaunch. Only one geometry set may be active at a time. Subsequent calls to rtuTraversalSetTriangles or rtuTraversalSetMesh will override any previously specified geometry. No internal copies of the triangle data are made. The user should ensure that the triangle data remains valid until after rtuTraversalTraverse has been called. Counter-clockwise winding is assumed for normal and backfacing computations.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |
| *num_tris* | Triangle count |
| *tris* | Triangles [ tri1_v1.x, tri1_v1.y, tr1_v1.z, tri1_v2.x, ... ] |

#### 3.5.4.14 RTresult RTAPI rtuTraversalTraverse ( RTUtraversal *traversal* )

Perform any necessary preprocessing (eg, acceleration structure building and kernel compilation ) and cast current rays against current geometry.

**Parameters**

| | |
|---|---|
| *traversal* | Traversal state handle |

#### 3.5.4.15 RTresult RTAPI rtuTraversalUnmapOutput ( RTUtraversal *traversal,* RTUoutput *which* )

See rtuTraversalMapOutput

#### 3.5.4.16 RTresult RTAPI rtuTraversalUnmapRays ( RTUtraversal *traversal* )

See rtuTraversalMapRays.

#### 3.5.4.17 RTresult RTAPI rtuTraversalUnmapResults ( RTUtraversal *traversal* )

See rtuTraversalMapResults

## 3.6 optixu_traversal.h

```


/******************************************************************************\
 *
 * Traversal API
 *
\******************************************************************************/

#ifndef _optixu_optux_traversal_h_
#define _optixu_optux_traversal_h_

#include "../optix.h"

#ifdef __cplusplus
```

```c
extern "C" {
#endif

  typedef struct {
    int   prim_id;
    float t;
  } RTUtraversalresult;


  typedef enum {
    RTU_QUERY_TYPE_ANY_HIT = 0,
    RTU_QUERY_TYPE_CLOSEST_HIT,
    RTU_QUERY_TYPE_COUNT
  } RTUquerytype;

  typedef enum  {
    RTU_RAYFORMAT_ORIGIN_DIRECTION_TMIN_TMAX_INTERLEAVED = 0,
    RTU_RAYFORMAT_ORIGIN_DIRECTION_INTERLEAVED,
    RTU_RAYFORMAT_COUNT
  } RTUrayformat;

  typedef enum  {
    RTU_TRIFORMAT_MESH= 0,
    RTU_TRIFORMAT_TRIANGLE_SOUP,
    RTU_TRIFORMAT_COUNT
  } RTUtriformat;

  typedef enum  {
    RTU_INITOPTION_NONE          = 0,
    RTU_INITOPTION_GPU_ONLY      = 1 << 0,
    RTU_INITOPTION_CPU_ONLY      = 1 << 1,
    RTU_INITOPTION_CULL_BACKFACE = 1 << 2
  } RTUinitoptions;

  typedef enum  {
    RTU_OUTPUT_NONE        = 0,
    RTU_OUTPUT_NORMAL      = 1 << 0, /*< float3 [x, y, z] */
    RTU_OUTPUT_BARYCENTRIC = 1 << 1, /*< float2 [alpha, beta] (gamma implicit) */
    RTU_OUTPUT_BACKFACING  = 1 << 2  /*< char   [1 | 0] */
  } RTUoutput;

  typedef enum  {
    RTU_OPTION_INT_NUM_THREADS=0
  } RTUoption;


  typedef struct RTUtraversal_api*  RTUtraversal;


  RTresult RTAPI rtuTraversalCreate( RTUtraversal* traversal,
                                     RTUquerytype  query_type,
                                     RTUrayformat  ray_format,
                                     RTUtriformat  tri_format,
                                     unsigned int  outputs,
                                     unsigned int  options,
                                     RTcontext     context );

  RTresult RTAPI rtuTraversalGetErrorString( RTUtraversal traversal,
                                             RTresult code,
                                             const char** return_string);
  RTresult RTAPI rtuTraversalSetOption( RTUtraversal traversal,
                                        RTUoption    option,
                                        void*        value );

  RTresult RTAPI rtuTraversalSetMesh( RTUtraversal    traversal,
                                      unsigned int    num_verts,
                                      const float*    verts,
                                      unsigned int    num_tris,
                                      const unsigned* indices );

  RTresult RTAPI rtuTraversalSetTriangles( RTUtraversal traversal,
                                               unsigned int num_tris,
                                               const float* tris );

  RTresult RTAPI rtuTraversalSetAccelData( RTUtraversal traversal,
                                           const void*  data,
                                           RTsize       data_size );

  RTresult RTAPI rtuTraversalGetAccelDataSize( RTUtraversal traversal,
                                               RTsize*      data_size );

  RTresult RTAPI rtuTraversalGetAccelData( RTUtraversal traversal,
                                           void*        data );

  RTresult RTAPI rtuTraversalMapRays( RTUtraversal traversal,
                                      unsigned int num_rays,
                                      float** rays );

  RTresult RTAPI rtuTraversalUnmapRays( RTUtraversal traversal );

  RTresult RTAPI rtuTraversalPreprocess( RTUtraversal traversal );

  RTresult RTAPI rtuTraversalTraverse( RTUtraversal traversal );

  RTresult RTAPI rtuTraversalMapResults( RTUtraversal         traversal,
                                         RTUtraversalresult** results );

  RTresult RTAPI rtuTraversalUnmapResults( RTUtraversal         traversal );

  RTresult RTAPI rtuTraversalMapOutput( RTUtraversal traversal,
                                        RTUoutput    which,
                                        void**       output );
  RTresult RTAPI rtuTraversalUnmapOutput( RTUtraversal traversal,
                                          RTUoutput    which );
  RTresult RTAPI rtuTraversalDestroy( RTUtraversal traversal );

#ifdef __cplusplus
} /* extern "C" */
#endif

#endif /* _optixu_optux_traversal.h */

```