OptiX Utility Library

2.1.0

Generated by Doxygen 1.6.3

Fri Jun 10 16:34:48 2011

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 113 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_INITOPTION_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 86 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 104 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 93 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.

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 46 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 55 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 67 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

→ ***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

→ ***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

→ ***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

→ ***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

→ ***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

→ ***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::operator[ ] (const std::string &varname)
- Handle< VariableObj > optix::Handle::operator[ ] (const char *varname)
- virtual void optix::APIObj::checkError (RTresult code)
- void optix::APIObj::checkErrorNoGetContext (RTresult code)
- Context optix::ContextObj::getContext ()
- static unsigned int optix::ContextObj::getDeviceCount ()
- static Context optix::ContextObj::create ()
- void optix::ContextObj::destroy ()
- void optix::ContextObj::validate ()
- void optix::ContextObj::compile ()
- int optix::ContextObj::getRunningState ()
- RTcontext optix::ContextObj::get ()
- void optix::ProgramObj::destroy ()
- void optix::ProgramObj::validate ()
- Context optix::ProgramObj::getContext ()
- Variable optix::ProgramObj::declareVariable (const std::string &name)
- Variable optix::ProgramObj::queryVariable (const std::string &name)
- void optix::ProgramObj::removeVariable (Variable v)
- unsigned int optix::ProgramObj::getVariableCount ()
- Variable optix::ProgramObj::getVariable (unsigned int index)
- RTprogram optix::ProgramObj::get ()
- void optix::GroupObj::destroy ()
- void optix::GroupObj::validate ()
- Context optix::GroupObj::getContext ()
- void optix::SelectorObj::destroy ()
- void optix::SelectorObj::validate ()
- Context optix::SelectorObj::getContext ()
- RTselector optix::SelectorObj::get ()
- RTgroup optix::GroupObj::get ()
- void optix::GeometryGroupObj::destroy ()
- void optix::GeometryGroupObj::validate ()
- Context optix::GeometryGroupObj::getContext ()
- RTgeometrygroup optix::GeometryGroupObj::get ()
- void optix::TransformObj::destroy ()
- void optix::TransformObj::validate ()
- Context optix::TransformObj::getContext ()
- RTtransform optix::TransformObj::get ()
- void optix::AccelerationObj::destroy ()
- void optix::AccelerationObj::validate ()
- Context optix::AccelerationObj::getContext ()
- RTacceleration optix::AccelerationObj::get ()
- void optix::GeometryInstanceObj::destroy ()
- void optix::GeometryInstanceObj::validate ()
- Context optix::GeometryInstanceObj::getContext ()
- RTgeometryinstance optix::GeometryInstanceObj::get ()
- void optix::GeometryObj::destroy ()

- void optix::GeometryObj::validate ()
- Context optix::GeometryObj::getContext ()
- RTgeometry optix::GeometryObj::get ()
- void optix::MaterialObj::destroy ()
- void optix::MaterialObj::validate ()
- Context optix::MaterialObj::getContext ()
- RTmaterial optix::MaterialObj::get ()
- void optix::TextureSamplerObj::destroy ()
- void optix::TextureSamplerObj::validate ()
- Context optix::TextureSamplerObj::getContext ()
- RTtexturesampler optix::TextureSamplerObj::get ()
- void optix::BufferObj::destroy ()
- void optix::BufferObj::validate ()
- Context optix::BufferObj::getContext ()
- RTbuffer optix::BufferObj::get ()
- Context optix::VariableObj::getContext ()
- std::string optix::VariableObj::getName ()
- std::string optix::VariableObj::getAnnotation ()
- RTobjecttype optix::VariableObj::getType ()
- RTvariable optix::VariableObj::get ()
- RTsize optix::VariableObj::getSize ()

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

- 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::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 ()
- unsigned int optix::ContextObj::getEnabledDeviceCount ()

- int optix::ContextObj::getMaxTextureCount ()
- RTsize optix::ContextObj::getAvailableDeviceMemory (int ordinal)

- void optix::ContextObj::setStackSize (RTsize stack_size_bytes)
- RTsize optix::ContextObj::getStackSize ()
- void optix::ContextObj::setEntryPointCount (unsigned int num_entry_points)
- unsigned int optix::ContextObj::getEntryPointCount ()
- void optix::ContextObj::setRayTypeCount (unsigned int num_ray_types)
- unsigned int optix::ContextObj::getRayTypeCount ()

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- Variable optix::GeometryObj::getVariable (unsigned int index)

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

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

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

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

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

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

- void optix::BufferObj::setFormat (RTformat format)
- RTformat optix::BufferObj::getFormat ()
- void optix::BufferObj::setElementSize (RTsize size_of_element)
- RTsize optix::BufferObj::getElementSize ()

- void optix::BufferObj::setSize (RTsize width)
- void optix::BufferObj::getSize (RTsize &width)
- void optix::BufferObj::setSize (RTsize width, RTsize height)
- void optix::BufferObj::getSize (RTsize &width, RTsize &height)
- void optix::BufferObj::setSize (RTsize width, RTsize height, RTsize depth)
- void optix::BufferObj::getSize (RTsize &width, RTsize &height, RTsize &depth)
- void optix::BufferObj::setSize (unsigned int dimensionality, const RTsize *dims)
- void optix::BufferObj::getSize (unsigned int dimensionality, RTsize *dims)
- unsigned int optix::BufferObj::getDimensionality ()

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

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

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::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)

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)

Numeric value getters

Query value of a variable with scalar numeric value

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

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)

User data variable accessors

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

OptiX API object getters

Reitrieve OptiX API object value from a variable

- Buffer optix::VariableObj::getBuffer ()
- TextureSampler optix::VariableObj::getTextureSampler ()

1.2.2 Typedef Documentation

1.2.2.1 typedef Handle<AccelerationObj> optix::Acceleration

Use this to manipulate RTacceleration objects.

Definition at line 194 of file optixpp_namespace.h.

1.2.2.2 typedef Handle<BufferObj> optix::Buffer

Use this to manipulate RTbuffer objects.

Definition at line 195 of file optixpp_namespace.h.

1.2.2.3 typedef Handle<ContextObj> optix::Context

Use this to manipulate RTcontext objects.

Definition at line 196 of file optixpp_namespace.h.

1.2.2.4 typedef Handle<GeometryObj> optix::Geometry

Use this to manipulate RTgeometry objects.

Definition at line 197 of file optixpp_namespace.h.

1.2.2.5 typedef Handle<GeometryGroupObj> optix::GeometryGroup

Use this to manipulate RTgeometrygroup objects.

Definition at line 198 of file optixpp_namespace.h.

1.2.2.6 typedef Handle<GeometryInstanceObj> optix::GeometryInstance

Use this to manipulate RTgeometryinstance objects.

Definition at line 199 of file optixpp_namespace.h.

1.2.2.7 typedef Handle<GroupObj> optix::Group

Use this to manipulate RTgroup objects.

Definition at line 200 of file optixpp_namespace.h.

1.2.2.8 typedef Handle<MaterialObj> optix::Material

Use this to manipulate RTmaterial objects.

Definition at line 201 of file optixpp_namespace.h.

1.2.2.9 typedef Handle<ProgramObj> optix::Program

Use this to manipulate RTprogram objects.

Definition at line 202 of file optixpp_namespace.h.

1.2.2.10 typedef Handle<SelectorObj> optix::Selector

Use this to manipulate RTselector objects.

Definition at line 203 of file optixpp_namespace.h.

1.2.2.11 typedef Handle<TextureSamplerObj> optix::TextureSampler

Use this to manipulate RTtexturesampler objects.

Definition at line 204 of file optixpp_namespace.h.

1.2.2.12 typedef Handle<TransformObj> optix::Transform

Use this to manipulate RTtransform objects.

Definition at line 205 of file optixpp_namespace.h.

1.2.2.13 typedef Handle<VariableObj> optix::Variable

Use this to manipulate RTvariable objects.

Definition at line 206 of file optixpp_namespace.h.

1.2.3 Function Documentation

1.2.3.1 unsigned int optix::GeometryInstanceObj::addMaterial (Material *material*) [inline, inherited]

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

Definition at line 2428 of file optixpp_namespace.h.

1.2.3.2 void optix::ContextObj::checkError (RTresult *code*) [inline, virtual, inherited]

See APIObj::checkError

Reimplemented from optix::APIObj.

Definition at line 1462 of file optixpp_namespace.h.

1.2.3.3 void optix::APIObj::checkError (RTresult *code*) [inline, virtual, inherited]

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 1442 of file optixpp_namespace.h.

1.2.3.4 void optix::APIObj::checkErrorNoGetContext (RTresult *code*) [inline, inherited]

Definition at line 1450 of file optixpp_namespace.h.

1.2.3.5 void optix::ContextObj::compile () [inline, inherited]

See rtContextCompile.

Definition at line 1872 of file optixpp_namespace.h.

1.2.3.6 Context optix::ContextObj::create () [inline, static, inherited]

Creates a Context object. See rtContextCreate.

Definition at line 1478 of file optixpp_namespace.h.

1.2.3.7 Acceleration optix::ContextObj::createAcceleration (const char * *builder*, const char * *traverser*) [inline, inherited]

See rtAccelerationCreate

Definition at line 1498 of file optixpp_namespace.h.

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

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

Definition at line 1541 of file optixpp_namespace.h.

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

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

Definition at line 1532 of file optixpp_namespace.h.

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

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

Definition at line 1523 of file optixpp_namespace.h.

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

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

Definition at line 1515 of file optixpp_namespace.h.

1.2.3.12 Buffer optix::ContextObj::createBuffer (unsigned int *type*) [inline, inherited]

Create a buffer with given RTbuffertype. See rtBufferCreate.

Definition at line 1508 of file optixpp_namespace.h.

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

Create buffer from GL buffer object. See rtBufferCreateFromGLBO.

Definition at line 1550 of file optixpp_namespace.h.

1.2.3.14 Geometry optix::ContextObj::createGeometry () [inline, inherited]

See rtGeometryCreate.

Definition at line 1625 of file optixpp_namespace.h.

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

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

Definition at line 1683 of file optixpp_namespace.h.

1.2.3.16 GeometryGroup optix::ContextObj::createGeometryGroup () [inline, inherited]

See rtGeometryGroupCreate.

Definition at line 1675 of file optixpp_namespace.h.

1.2.3.17 template<class Iterator > GeometryInstance optix::ContextObj::createGeometryInstance (Geometry *geometry*, Iterator *matlbegin*, Iterator *matlend*) [inline, inherited]

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

Definition at line 1640 of file optixpp_namespace.h.

1.2.3.18 GeometryInstance optix::ContextObj::createGeometryInstance () [inline, inherited]

See rtGeometryInstanceCreate.

Definition at line 1632 of file optixpp_namespace.h.

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

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

Definition at line 1662 of file optixpp_namespace.h.

1.2.3.20 Group optix::ContextObj::createGroup () [inline, inherited]

See rtGroupCreate.

Definition at line 1654 of file optixpp_namespace.h.

1.2.3.21 Material optix::ContextObj::createMaterial () [inline, inherited]

See rtMaterialCreate.

Definition at line 1703 of file optixpp_namespace.h.

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

See rtProgramCreateFromPTXFile.

Definition at line 1710 of file optixpp_namespace.h.

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

See rtProgramCreateFromPTXString.

Definition at line 1717 of file optixpp_namespace.h.

1.2.3.24 Selector optix::ContextObj::createSelector () [inline, inherited]

See rtSelectorCreate.

Definition at line 1724 of file optixpp_namespace.h.

1.2.3.25 TextureSampler optix::ContextObj::createTextureSampler () [inline, inherited]

See rtTextureSamplerCreate.

Definition at line 1731 of file optixpp_namespace.h.

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

Create TextureSampler from GL image. See rtTextureSamplerCreateFromGLImage.

Definition at line 1618 of file optixpp_namespace.h.

1.2.3.27 Transform optix::ContextObj::createTransform () [inline, inherited]

See rtTransformCreate.

Definition at line 1696 of file optixpp_namespace.h.

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

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 2618 of file optixpp_namespace.h.

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

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 2527 of file optixpp_namespace.h.

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

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 2436 of file optixpp_namespace.h.

1.2.3.31 Variable optix::SelectorObj::declareVariable (const std::string & *name*) [inline, inherited]

Definition at line 2102 of file optixpp_namespace.h.

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

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 1992 of file optixpp_namespace.h.

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

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 1936 of file optixpp_namespace.h.

1.2.3.34 void optix::BufferObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2816 of file optixpp_namespace.h.

1.2.3.35 void optix::TextureSamplerObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2656 of file optixpp_namespace.h.

1.2.3.36 void optix::MaterialObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2577 of file optixpp_namespace.h.

1.2.3.37 void optix::GeometryObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2474 of file optixpp_namespace.h.

1.2.3.38 void optix::GeometryInstanceObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2374 of file optixpp_namespace.h.

1.2.3.39 void optix::AccelerationObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2287 of file optixpp_namespace.h.

1.2.3.40 void optix::TransformObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2241 of file optixpp_namespace.h.

1.2.3.41 void optix::GeometryGroupObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2183 of file optixpp_namespace.h.

1.2.3.42 void optix::SelectorObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2047 of file optixpp_namespace.h.

1.2.3.43 void optix::GroupObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2030 of file optixpp_namespace.h.

1.2.3.44 void optix::ProgramObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 1975 of file optixpp_namespace.h.

1.2.3.45 void optix::ContextObj::destroy () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 1487 of file optixpp_namespace.h.

1.2.3.46 RTvariable optix::VariableObj::get () [inline, inherited]

Get the OptiX C API object wrapped by this instance.

Definition at line 3297 of file optixpp_namespace.h.

1.2.3.47 RTbuffer optix::BufferObj::get () [inline, inherited]

Get the underlying OptiX C API RTbuffer opaque pointer.

Definition at line 2989 of file optixpp_namespace.h.

1.2.3.48 RTtexturesampler optix::TextureSamplerObj::get () [inline, inherited]

Get the underlying OptiX C API RTtexturesampler opaque pointer.

Definition at line 2767 of file optixpp_namespace.h.

1.2.3.49 RTmaterial optix::MaterialObj::get () [inline, inherited]

Get the underlying OptiX C API RTmaterial opaque pointer.

Definition at line 2651 of file optixpp_namespace.h.

1.2.3.50 RTgeometry optix::GeometryObj::get () [inline, inherited]

Get the underlying OptiX C API RTgeometry opaque pointer.

Definition at line 2572 of file optixpp_namespace.h.

1.2.3.51 RTgeometryinstance optix::GeometryInstanceObj::get () [inline, inherited]

Get the underlying OptiX C API RTgeometryinstance opaque pointer.

Definition at line 2469 of file optixpp_namespace.h.

1.2.3.52 RTacceleration optix::AccelerationObj::get () [inline, inherited]

Get the underlying OptiX C API RTacceleration opaque pointer.

Definition at line 2369 of file optixpp_namespace.h.

1.2.3.53 RTtransform optix::TransformObj::get () [inline, inherited]

Get the underlying OptiX C API RTtransform opaque pointer.

Definition at line 2282 of file optixpp_namespace.h.

1.2.3.54 RTgeometrygroup optix::GeometryGroupObj::get () [inline, inherited]

Get the underlying OptiX C API RTgeometrygroup opaque pointer.

Definition at line 2236 of file optixpp_namespace.h.

1.2.3.55 RTgroup optix::GroupObj::get () [inline, inherited]

Get the underlying OptiX C API RTgroup opaque pointer.

Definition at line 2178 of file optixpp_namespace.h.

1.2.3.56 RTselector optix::SelectorObj::get () [inline, inherited]

Get the underlying OptiX C API RTselector opaque pointer.

Definition at line 2135 of file optixpp_namespace.h.

1.2.3.57 RTprogram optix::ProgramObj::get () [inline, inherited]

Definition at line 2025 of file optixpp_namespace.h.

1.2.3.58 RTcontext optix::ContextObj::get () [inline, inherited]

Return the OptiX C API RTcontext object.

Definition at line 1970 of file optixpp_namespace.h.

1.2.3.59 Acceleration optix::GeometryGroupObj::getAcceleration () [inline, inherited]

Query the Acceleration structure for this group. See rtGeometryGroupGetAcceleration.

Definition at line 2205 of file optixpp_namespace.h.

1.2.3.60 Acceleration optix::GroupObj::getAcceleration () [inline, inherited]

Query the Acceleration structure for this group. See rtGroupGetAcceleration.

Definition at line 2145 of file optixpp_namespace.h.

1.2.3.61 std::string optix::VariableObj::getAnnotation () [inline, inherited]

Retrieve the annotation associated with the variable.

Definition at line 3283 of file optixpp_namespace.h.

1.2.3.62 Program optix::MaterialObj::getAnyHitProgram (unsigned int *ray_type_index*) [inline, inherited]

Get any hit program for this material at the given *ray_type* index. See rtMaterialGetAnyHitProgram.

Definition at line 2611 of file optixpp_namespace.h.

1.2.3.63 unsigned int optix::TextureSamplerObj::getArraySize () [inline, inherited]

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

Definition at line 2690 of file optixpp_namespace.h.

1.2.3.64 RTsize optix::ContextObj::getAvailableDeviceMemory (int *ordinal*) [inline, inherited]

See rtContextGetAttribute.

Definition at line 1775 of file optixpp_namespace.h.

1.2.3.65 Program optix::GeometryObj::getBoundingBoxProgram () [inline, inherited]

Get the bounding box program for this geometry. See rtGeometryGetBoundingBoxProgram.

Definition at line 2508 of file optixpp_namespace.h.

1.2.3.66 Buffer optix::VariableObj::getBuffer () [inline, inherited]

Definition at line 3268 of file optixpp_namespace.h.

1.2.3.67 Buffer optix::TextureSamplerObj::getBuffer (unsigned int *texture_array_idx*, unsigned int *mip_level*) [inline, inherited]

Get the underlying buffer used for texture storage. rtTextureSamplerGetBuffer.

Definition at line 2760 of file optixpp_namespace.h.

1.2.3.68 std::string optix::AccelerationObj::getBuilder () [inline, inherited]

Query the acceleration structure builder. See rtAccelerationGetBuilder.

Definition at line 2333 of file optixpp_namespace.h.

1.2.3.69 template<typename T > T optix::TransformObj::getChild () [inline, inherited]

Set the child node of this transform. See rtTransformGetChild.

Definition at line 2265 of file optixpp_namespace.h.

1.2.3.70 GeometryInstance optix::GeometryGroupObj::getChild (unsigned int *index*) [inline, inherited]

Query an indexed GeometryInstance within this group. See rtGeometryGroupGetChild.

Definition at line 2229 of file optixpp_namespace.h.

1.2.3.71 template<typename T > T optix::GroupObj::getChild (unsigned int *index*) [inline, inherited]

Query an indexed child within this group. See rtGroupGetChild.

Definition at line 2171 of file optixpp_namespace.h.

1.2.3.72 template<typename T > T optix::SelectorObj::getChild (unsigned int *index*) [inline, inherited]

Query an indexed child within this group. See rtSelectorGetChild.

Definition at line 2095 of file optixpp_namespace.h.

1.2.3.73 unsigned int optix::GeometryGroupObj::getChildCount () [inline, inherited]

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

Definition at line 2217 of file optixpp_namespace.h.

1.2.3.74 unsigned int optix::GroupObj::getChildCount () [inline, inherited]

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

Definition at line 2157 of file optixpp_namespace.h.

1.2.3.75 unsigned int optix::SelectorObj::getChildCount () [inline, inherited]

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

Definition at line 2081 of file optixpp_namespace.h.

1.2.3.76 Program optix::MaterialObj::getClosestHitProgram (unsigned int *ray_type_index*) [inline, inherited]

Get closest hit program for this material at the given *ray_type* index. See rtMaterialGetClosestHitProgram.

Definition at line 2599 of file optixpp_namespace.h.

1.2.3.77 Context optix::VariableObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 2994 of file optixpp_namespace.h.

1.2.3.78 Context optix::BufferObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 2826 of file optixpp_namespace.h.

1.2.3.79 Context optix::TextureSamplerObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 2666 of file optixpp_namespace.h.

1.2.3.80 Context optix::MaterialObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 2587 of file optixpp_namespace.h.

1.2.3.81 Context optix::GeometryObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 2484 of file optixpp_namespace.h.

1.2.3.82 Context optix::GeometryInstanceObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 2384 of file optixpp_namespace.h.

1.2.3.83 Context optix::AccelerationObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 2297 of file optixpp_namespace.h.

1.2.3.84 Context optix::TransformObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 2251 of file optixpp_namespace.h.

1.2.3.85 Context optix::GeometryGroupObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 2193 of file optixpp_namespace.h.

1.2.3.86 Context optix::SelectorObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 2057 of file optixpp_namespace.h.

1.2.3.87 Context optix::GroupObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 2040 of file optixpp_namespace.h.

1.2.3.88 Context optix::ProgramObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 1985 of file optixpp_namespace.h.

1.2.3.89 Context optix::ContextObj::getContext () [inline, virtual, inherited]

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

Implements optix::APIObj.

Definition at line 1457 of file optixpp_namespace.h.

1.2.3.90 void optix::AccelerationObj::getData (void * *data*) [inline, inherited]

Get the marshalled acceleration data. See rtAccelerationGetData.

Definition at line 2359 of file optixpp_namespace.h.

1.2.3.91 RTsize optix::AccelerationObj::getDataSize () [inline, inherited]

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

Definition at line 2352 of file optixpp_namespace.h.

1.2.3.92 unsigned int optix::ContextObj::getDeviceCount () [inline, static, inherited]

Call rtDeviceGetDeviceCount and returns number of valid devices.

Definition at line 1468 of file optixpp_namespace.h.

1.2.3.93 unsigned int optix::BufferObj::getDimensionality () [inline, inherited]

Query dimensionality of buffer. See rtBufferGetDimensionality.

Definition at line 2897 of file optixpp_namespace.h.

1.2.3.94 RTsize optix::BufferObj::getElementSize () [inline, inherited]

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

Definition at line 2850 of file optixpp_namespace.h.

1.2.3.95 unsigned int optix::ContextObj::getEnabledDeviceCount () [inline, inherited]

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

Definition at line 1761 of file optixpp_namespace.h.

1.2.3.96 std::vector< int > optix::ContextObj::getEnabledDevices () [inline, inherited]

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

Definition at line 1753 of file optixpp_namespace.h.

1.2.3.97 unsigned int optix::ContextObj::getEntryPointCount () [inline, inherited]

See rtContextgetEntryPointCount.

Definition at line 1801 of file optixpp_namespace.h.

1.2.3.98 std::string optix::ContextObj::getErrorString (RTresult *code*) [inline, inherited]

See rtContextGetErrroString.

Definition at line 1738 of file optixpp_namespace.h.

1.2.3.99 bool optix::ContextObj::getExceptionEnabled (RTexception *exception*) [inline, inherited]

See rtContextGetExceptionEnabled.

Definition at line 1840 of file optixpp_namespace.h.

1.2.3.100 Program optix::ContextObj::getExceptionProgram (unsigned int *entry_point_index*) [inline, inherited]

See rtContextGetExceptionProgram.

Definition at line 1827 of file optixpp_namespace.h.

1.2.3.101 void optix::TextureSamplerObj::getFilteringModes (RTfiltermode & *minification*, RTfiltermode & *magnification*, RTfiltermode & *mipmapping*) [inline, inherited]

Query filtering modes for this sampler. See rtTextureSamplerGetFilteringModes.

Definition at line 2714 of file optixpp_namespace.h.

1.2.3.102 float optix::VariableObj::getFloat () [inline, inherited]

Definition at line 3197 of file optixpp_namespace.h.

1.2.3.103 RTformat optix::BufferObj::getFormat () [inline, inherited]

Query the data format for the buffer. See rtBufferGetFormat.

Definition at line 2838 of file optixpp_namespace.h.

1.2.3.104 Geometry optix::GeometryInstanceObj::getGeometry () [inline, inherited]

Get the geometry object associated with this instance. See rtGeometryInstanceGetGeometry.

Definition at line 2396 of file optixpp_namespace.h.

1.2.3.105 unsigned int optix::BufferObj::getGLBOId () [inline, inherited]

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

Definition at line 2904 of file optixpp_namespace.h.

1.2.3.106 RTtextureindexmode optix::TextureSamplerObj::getIndexingMode () [inline, inherited]

Query texture indexing mode for this sampler. See rtTextureSamplerGetIndexingMode.

Definition at line 2748 of file optixpp_namespace.h.

1.2.3.107 int optix::VariableObj::getInt () [inline, inherited]

Definition at line 3211 of file optixpp_namespace.h.

1.2.3.108 Program optix::GeometryObj::getIntersectionProgram () [inline, inherited]

Get the intersection program for this geometry. See rtGeometryGetIntersectionProgram.

Definition at line 2520 of file optixpp_namespace.h.

1.2.3.109 Material optix::GeometryInstanceObj::getMaterial (unsigned int *idx*) [inline, inherited]

Get the material at given index. See rtGeometryInstanceGetMaterial.

Definition at line 2420 of file optixpp_namespace.h.

1.2.3.110 unsigned int optix::GeometryInstanceObj::getMaterialCount () [inline, inherited]

Query the number of materials associated with this instance. See rtGeometryInstanceGetMaterialCount.

Definition at line 2408 of file optixpp_namespace.h.

1.2.3.111 void optix::TransformObj::getMatrix (bool *transpose*, float * *matrix*, float * *inverse_matrix*) [inline, inherited]

Get the transform matrix for this node. See rtTransformGetMatrix.

Definition at line 2277 of file optixpp_namespace.h.

1.2.3.112 float optix::TextureSamplerObj::getMaxAnisotropy () [inline, inherited]

Query maximum anisotropy for this sampler. See rtTextureSamplerGetMaxAnisotropy.

Definition at line 2724 of file optixpp_namespace.h.

1.2.3.113 int optix::ContextObj::getMaxTextureCount () [inline, inherited]

See rtContextGetAttribute

Definition at line 1768 of file optixpp_namespace.h.

1.2.3.114 unsigned int optix::TextureSamplerObj::getMipLevelCount () [inline, inherited]

Query the number of mip levels for this sampler. See rtTextureSamplerGetMipLevelCount.

Definition at line 2678 of file optixpp_namespace.h.

1.2.3.115 Program optix::ContextObj::getMissProgram (unsigned int *ray_type_index*) [inline, inherited]

See rtContextGetMissProgram.

Definition at line 1865 of file optixpp_namespace.h.

1.2.3.116 std::string optix::VariableObj::getName () [inline, inherited]

Retrieve the name of the variable.

Definition at line 3276 of file optixpp_namespace.h.

1.2.3.117 unsigned int optix::GeometryObj::getPrimitiveCount () [inline, inherited]

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

Definition at line 2496 of file optixpp_namespace.h.

1.2.3.118 RTsize optix::ContextObj::getPrintBufferSize () [inline, inherited]

See rtContextGetPrintBufferSize.

Definition at line 1917 of file optixpp_namespace.h.

1.2.3.119 bool optix::ContextObj::getPrintEnabled () [inline, inherited]

See rtContextGetPrintEnabled.

Definition at line 1905 of file optixpp_namespace.h.

1.2.3.120 optix::int3 optix::ContextObj::getPrintLaunchIndex () [inline, inherited]

See rtContextGetPrintLaunchIndex.

Definition at line 1929 of file optixpp_namespace.h.

1.2.3.121 std::string optix::AccelerationObj::getProperty (const std::string & *name*) [inline, inherited]

Query properties specifying Acceleration builder/traverser behavior. See rtAccelerationGetProperty.

Definition at line 2321 of file optixpp_namespace.h.

1.2.3.122 Program optix::ContextObj::getRayGenerationProgram (unsigned int *entry_point_index*) [inline, inherited]

See rtContextGetRayGenerationProgram.

Definition at line 1814 of file optixpp_namespace.h.

1.2.3.123 unsigned int optix::ContextObj::getRayTypeCount () [inline, inherited]

See rtContextGetRayTypeCount.

Definition at line 1853 of file optixpp_namespace.h.

1.2.3.124 RTtexturereadmode optix::TextureSamplerObj::getReadMode () [inline, inherited]

Query texture read mode for this sampler. See rtTextureSamplerGetReadMode.

Definition at line 2736 of file optixpp_namespace.h.

1.2.3.125 int optix::ContextObj::getRunningState () [inline, inherited]

See rtContextGetRunningState.

Definition at line 1893 of file optixpp_namespace.h.

1.2.3.126 RTsize optix::VariableObj::getSize () [inline, inherited]

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

Definition at line 3302 of file optixpp_namespace.h.

1.2.3.127 void optix::BufferObj::getSize (unsigned int *dimensionality*, RTsize ∗ *dims*) [inline, inherited]

Query dimensions of buffer. See rtBufferGetSizev.

Definition at line 2892 of file optixpp_namespace.h.

1.2.3.128 void optix::BufferObj::getSize (RTsize & *width*, RTsize & *height*, RTsize & *depth*) [inline, inherited]

Query 3D buffer dimension. See rtBufferGetSize3D.

Definition at line 2882 of file optixpp_namespace.h.

1.2.3.129 void optix::BufferObj::getSize (RTsize & *width*, RTsize & *height*) [inline, inherited]

Query 2D buffer dimension. See rtBufferGetSize2D.

Definition at line 2872 of file optixpp_namespace.h.

1.2.3.130 void optix::BufferObj::getSize (RTsize & *width*) [inline, inherited]

Query 1D buffer dimension. See rtBufferGetSize1D.

Definition at line 2862 of file optixpp_namespace.h.

1.2.3.131 RTsize optix::ContextObj::getStackSize () [inline, inherited]

See rtContextGetStackSize.

Definition at line 1789 of file optixpp_namespace.h.

1.2.3.132 optix::TextureSampler optix::VariableObj::getTextureSampler () [inline, inherited]

Definition at line 3309 of file optixpp_namespace.h.

1.2.3.133 std::string optix::AccelerationObj::getTraverser () [inline, inherited]

Query the acceleration structure traverser. See rtAccelerationGetTraverser.

Definition at line 2345 of file optixpp_namespace.h.

1.2.3.134 RTobjecttype optix::VariableObj::getType () [inline, inherited]

Query the object type of the variable.

Definition at line 3290 of file optixpp_namespace.h.

1.2.3.135 unsigned int optix::VariableObj::getUint () [inline, inherited]

Definition at line 3204 of file optixpp_namespace.h.

1.2.3.136 void optix::VariableObj::getUserData (RTsize *size*, void * *ptr*) [inline, inherited]

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

Definition at line 3233 of file optixpp_namespace.h.

1.2.3.137 Variable optix::MaterialObj::getVariable (unsigned int *index*) [inline, virtual, inherited]

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

Implements optix::ScopedObj.

Definition at line 2644 of file optixpp_namespace.h.

1.2.3.138 Variable optix::GeometryObj::getVariable (unsigned int *index*) [inline, virtual, inherited]

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

Implements optix::ScopedObj.

Definition at line 2553 of file optixpp_namespace.h.

1.2.3.139 Variable optix::GeometryInstanceObj::getVariable (unsigned int *index*) [inline, virtual, inherited]

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

Implements optix::ScopedObj.

Definition at line 2462 of file optixpp_namespace.h.

1.2.3.140 Variable optix::SelectorObj::getVariable (unsigned int *index*) [inline, inherited]

Definition at line 2128 of file optixpp_namespace.h.

1.2.3.141 Variable optix::ProgramObj::getVariable (unsigned int *index*) [inline, virtual, inherited]

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

Implements optix::ScopedObj.

Definition at line 2018 of file optixpp_namespace.h.

1.2.3.142 Variable optix::ContextObj::getVariable (unsigned int *index*) [inline, virtual, inherited]

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

Implements optix::ScopedObj.

Definition at line 1962 of file optixpp_namespace.h.

1.2.3.143 unsigned int optix::MaterialObj::getVariableCount () [inline, virtual, inherited]

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 2637 of file optixpp_namespace.h.

1.2.3.144 unsigned int optix::GeometryObj::getVariableCount () [inline, virtual, inherited]

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 2546 of file optixpp_namespace.h.

1.2.3.145 unsigned int optix::GeometryInstanceObj::getVariableCount () [inline, virtual, inherited]

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 2455 of file optixpp_namespace.h.

1.2.3.146 unsigned int optix::SelectorObj::getVariableCount () [inline, inherited]

Definition at line 2121 of file optixpp_namespace.h.

1.2.3.147 unsigned int optix::ProgramObj::getVariableCount () [inline, virtual, inherited]

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 2011 of file optixpp_namespace.h.

1.2.3.148 unsigned int optix::ContextObj::getVariableCount () [inline, virtual, inherited]

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 1955 of file optixpp_namespace.h.

1.2.3.149 Program optix::SelectorObj::getVisitProgram () [inline, inherited]

Get the visitor program for this selector. See rtSelectorGetVisitProgram.

Definition at line 2069 of file optixpp_namespace.h.

1.2.3.150 RTwrapmode optix::TextureSamplerObj::getWrapMode (unsigned int *dim*) [inline, inherited]

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

Definition at line 2702 of file optixpp_namespace.h.

1.2.3.151 bool optix::GeometryObj::isDirty () [inline, inherited]

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

Definition at line 2565 of file optixpp_namespace.h.

1.2.3.152 bool optix::AccelerationObj::isDirty () [inline, inherited]

Query if the acceleration needs a rebuild. See rtAccelerationIsDirty.

Definition at line 2309 of file optixpp_namespace.h.

1.2.3.153 void optix::ContextObj::launch (unsigned int *entry_point_index*, RTsize *image_width*, RTsize *image_height*, RTsize *image_depth*) [inline, inherited]

See rtContextLaunch3D.

Definition at line 1887 of file optixpp_namespace.h.

1.2.3.154 void optix::ContextObj::launch (unsigned int *entry_point_index*, RTsize *image_width*, RTsize *image_height*) [inline, inherited]

See rtContextLaunch2D.

Definition at line 1882 of file optixpp_namespace.h.

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

See rtContextLaunch1D

Definition at line 1877 of file optixpp_namespace.h.

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

For backwards compatability. Use Exception::makeException instead.

Definition at line 299 of file optixpp_namespace.h.

1.2.3.157 Exception optix::Exception::makeException (RTresult *code*, RTcontext *context*) [inline, static, inherited]

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

Definition at line 245 of file optixpp_namespace.h.

1.2.3.158 void * optix::BufferObj::map () [inline, inherited]

Maps a buffer object for host access. See rtBufferMap.

Definition at line 2976 of file optixpp_namespace.h.

1.2.3.159 void optix::GeometryObj::markDirty () [inline, inherited]

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

Definition at line 2560 of file optixpp_namespace.h.

1.2.3.160 void optix::AccelerationObj::markDirty () [inline, inherited]

Mark the acceleration as needing a rebuild. See rtAccelerationMarkDirty.

Definition at line 2304 of file optixpp_namespace.h.

1.2.3.161 template<class T > Handle< VariableObj > optix::Handle< T >::operator[ ] (const char * *varname*) [inline, inherited]

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 584 of file optixpp_namespace.h.

1.2.3.162 template<class T > Handle< VariableObj > optix::Handle< T >::operator[ ] (const std::string & *varname*) [inline, inherited]

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 575 of file optixpp_namespace.h.

1.2.3.163 Variable optix::MaterialObj::queryVariable (const std::string & *name*) [inline, virtual, inherited]

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 2625 of file optixpp_namespace.h.

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

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 2534 of file optixpp_namespace.h.

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

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 2443 of file optixpp_namespace.h.

1.2.3.166 Variable optix::SelectorObj::queryVariable (const std::string & *name*) [inline, inherited]

Definition at line 2109 of file optixpp_namespace.h.

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

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 1999 of file optixpp_namespace.h.

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

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 1943 of file optixpp_namespace.h.

1.2.3.169 void optix::BufferObj::registerGLBuffer () [inline, inherited]

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

Definition at line 2911 of file optixpp_namespace.h.

1.2.3.170 void optix::TextureSamplerObj::registerGLTexture () [inline, inherited]

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

Definition at line 2772 of file optixpp_namespace.h.

1.2.3.171 void optix::MaterialObj::removeVariable (Variable *v*) [inline, virtual, inherited]

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2632 of file optixpp_namespace.h.

1.2.3.172 void optix::GeometryObj::removeVariable (Variable *v*) [inline, virtual, inherited]

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2541 of file optixpp_namespace.h.

1.2.3.173 void optix::GeometryInstanceObj::removeVariable (Variable *v*) [inline, virtual, inherited]

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2450 of file optixpp_namespace.h.

1.2.3.174 void optix::SelectorObj::removeVariable (Variable *v*) [inline, inherited]

Definition at line 2116 of file optixpp_namespace.h.

1.2.3.175 void optix::ProgramObj::removeVariable (Variable *v*) [inline, virtual, inherited]

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 2006 of file optixpp_namespace.h.

1.2.3.176 void optix::ContextObj::removeVariable (Variable *v*) [inline, virtual, inherited]

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 1950 of file optixpp_namespace.h.

1.2.3.177 void optix::VariableObj::set (Buffer *buffer*) [inline, inherited]

Definition at line 3223 of file optixpp_namespace.h.

1.2.3.178 void optix::VariableObj::set1fv (const float * *f*) [inline, inherited]

Set variable value to a scalar float.

Definition at line 3121 of file optixpp_namespace.h.

1.2.3.179 void optix::VariableObj::set1iv (const int * *i*) [inline, inherited]

Definition at line 3177 of file optixpp_namespace.h.

1.2.3.180 void optix::VariableObj::set1uiv (const unsigned int * *u*) [inline, inherited]

Definition at line 3021 of file optixpp_namespace.h.

1.2.3.181 void optix::VariableObj::set2fv (const float * f) [inline, inherited]

Set variable value to a float2.

Definition at line 3126 of file optixpp_namespace.h.

1.2.3.182 void optix::VariableObj::set2iv (const int * i) [inline, inherited]

Definition at line 3182 of file optixpp_namespace.h.

1.2.3.183 void optix::VariableObj::set2uiv (const unsigned int * u) [inline, inherited]

Definition at line 3026 of file optixpp_namespace.h.

1.2.3.184 void optix::VariableObj::set3fv (const float * f) [inline, inherited]

Set variable value to a float3.

Definition at line 3131 of file optixpp_namespace.h.

1.2.3.185 void optix::VariableObj::set3iv (const int * i) [inline, inherited]

Definition at line 3187 of file optixpp_namespace.h.

1.2.3.186 void optix::VariableObj::set3uiv (const unsigned int * u) [inline, inherited]

Definition at line 3031 of file optixpp_namespace.h.

1.2.3.187 void optix::VariableObj::set4fv (const float * f) [inline, inherited]

Set variable value to a float4.

Definition at line 3136 of file optixpp_namespace.h.

1.2.3.188 void optix::VariableObj::set4iv (const int * i) [inline, inherited]

Definition at line 3192 of file optixpp_namespace.h.

1.2.3.189 void optix::VariableObj::set4uiv (const unsigned int * *u*) [inline, inherited]

Definition at line 3036 of file optixpp_namespace.h.

1.2.3.190 void optix::GeometryGroupObj::setAcceleration (Acceleration *acceleration*) [inline, inherited]

Set the Acceleration structure for this group. See rtGeometryGroupSetAcceleration.

Definition at line 2200 of file optixpp_namespace.h.

1.2.3.191 void optix::GroupObj::setAcceleration (Acceleration *acceleration*) [inline, inherited]

Set the Acceleration structure for this group. See rtGroupSetAcceleration.

Definition at line 2140 of file optixpp_namespace.h.

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

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

Definition at line 2606 of file optixpp_namespace.h.

1.2.3.193 void optix::TextureSamplerObj::setArraySize (unsigned int *num_textures_in_array*) [inline, inherited]

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

Definition at line 2685 of file optixpp_namespace.h.

1.2.3.194 void optix::GeometryObj::setBoundingBoxProgram (Program *program*) [inline, inherited]

Set the bounding box program for this geometry. See rtGeometrySetBoundingBoxProgram.

Definition at line 2503 of file optixpp_namespace.h.

1.2.3.195 void optix::VariableObj::setBuffer (Buffer *buffer*) [inline, inherited]

Definition at line 3218 of file optixpp_namespace.h.

1.2.3.196 void optix::TextureSamplerObj::setBuffer (unsigned int *texture_array_idx*, unsigned int *mip_level*, Buffer *buffer*) [inline, inherited]

Set the underlying buffer used for texture storage. rtTextureSamplerSetBuffer.

Definition at line 2755 of file optixpp_namespace.h.

1.2.3.197 void optix::AccelerationObj::setBuilder (const std::string & *builder*) [inline, inherited]

Specify the acceleration structure builder. See rtAccelerationSetBuilder.

Definition at line 2328 of file optixpp_namespace.h.

1.2.3.198 template<typename T > void optix::TransformObj::setChild (T *child*) [inline, inherited]

Set the child node of this transform. See rtTransformSetChild.

Definition at line 2259 of file optixpp_namespace.h.

1.2.3.199 void optix::GeometryGroupObj::setChild (unsigned int *index*, GeometryInstance *geometryinstance*) [inline, inherited]

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

Definition at line 2224 of file optixpp_namespace.h.

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

Set an indexed child within this group. See rtGroupSetChild.

Definition at line 2165 of file optixpp_namespace.h.

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

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

Definition at line 2089 of file optixpp_namespace.h.

1.2.3.202 void optix::GeometryGroupObj::setChildCount (unsigned int *count*) [inline, inherited]

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

Definition at line 2212 of file optixpp_namespace.h.

1.2.3.203 void optix::GroupObj::setChildCount (unsigned int *count*) [inline, inherited]

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

Definition at line 2152 of file optixpp_namespace.h.

1.2.3.204 void optix::SelectorObj::setChildCount (unsigned int *count*) [inline, inherited]

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

Definition at line 2076 of file optixpp_namespace.h.

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

Set closest hit program for this material at the given *ray_type* index. See rtMaterialSetClosestHitProgram.

Definition at line 2594 of file optixpp_namespace.h.

1.2.3.206 void optix::AccelerationObj::setData (const void * *data*, RTsize *size*) [inline, inherited]

Specify the acceleration structure via marshalled acceleration data. See rtAccelerationSetData.

Definition at line 2364 of file optixpp_namespace.h.

1.2.3.207 template<class Iterator > void optix::ContextObj::setDevices (Iterator *begin*, Iterator *end*) [inline, inherited]

See rtContextSetDevices

Definition at line 1746 of file optixpp_namespace.h.

1.2.3.208 void optix::BufferObj::setElementSize (RTsize *size_of_element*) [inline, inherited]

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

Definition at line 2845 of file optixpp_namespace.h.

1.2.3.209 void optix::ContextObj::setEntryPointCount (unsigned int *num_entry_points*) [inline, inherited]

See rtContextSetEntryPointCount.

Definition at line 1796 of file optixpp_namespace.h.

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

See rtContextSetExceptionEnabled.

Definition at line 1835 of file optixpp_namespace.h.

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

See rtContextSetExceptionProgram.

Definition at line 1822 of file optixpp_namespace.h.

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

Set filtering modes for this sampler. See rtTextureSamplerSetFilteringModes.

Definition at line 2709 of file optixpp_namespace.h.

1.2.3.213 void optix::VariableObj::setFloat (float *f1*, float *f2*, float *f3*, float *f4*) [inline, inherited]

Set variable value to a float4.

Definition at line 3116 of file optixpp_namespace.h.

1.2.3.214 void optix::VariableObj::setFloat (optix::float4 *f*) [inline, inherited]

Set variable value to a float4.

Definition at line 3111 of file optixpp_namespace.h.

1.2.3.215 void optix::VariableObj::setFloat (float *f1*, float *f2*, float *f3*) [inline, inherited]

Set variable value to a float3.

Definition at line 3106 of file optixpp_namespace.h.

1.2.3.216 void optix::VariableObj::setFloat (optix::float3 *f*) [inline, inherited]

Set variable value to a float3.

Definition at line 3101 of file optixpp_namespace.h.

1.2.3.217 void optix::VariableObj::setFloat (float *f1*, float *f2*) [inline, inherited]

Set variable value to a float2.

Definition at line 3096 of file optixpp_namespace.h.

1.2.3.218 void optix::VariableObj::setFloat (optix::float2 *f*) [inline, inherited]

Set variable value to a float2.

Definition at line 3091 of file optixpp_namespace.h.

1.2.3.219 void optix::VariableObj::setFloat (float *f1*) [inline, inherited]

Set variable value to a scalar float.

Definition at line 3086 of file optixpp_namespace.h.

1.2.3.220 void optix::BufferObj::setFormat (RTformat *format*) [inline, inherited]

Set the data format for the buffer. See rtBufferSetFormat.

Definition at line 2833 of file optixpp_namespace.h.

1.2.3.221 void optix::GeometryInstanceObj::setGeometry (Geometry *geometry*) [inline, inherited]

Set the geometry object associated with this instance. See rtGeometryInstanceSetGeometry.

Definition at line 2391 of file optixpp_namespace.h.

1.2.3.222 void optix::TextureSamplerObj::setIndexingMode (RTtextureindexmode *indexmode*) [inline, inherited]

Set texture indexing mode for this sampler. See rtTextureSamplerSetIndexingMode.

Definition at line 2743 of file optixpp_namespace.h.

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

Definition at line 3172 of file optixpp_namespace.h.

1.2.3.224 void optix::VariableObj::setInt (optix::int4 *i*) [inline, inherited]

Definition at line 3167 of file optixpp_namespace.h.

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

Definition at line 3162 of file optixpp_namespace.h.

1.2.3.226 void optix::VariableObj::setInt (optix::int3 *i*) [inline, inherited]

Definition at line 3157 of file optixpp_namespace.h.

1.2.3.227 void optix::VariableObj::setInt (int *i1*, int *i2*) [inline, inherited]

Definition at line 3152 of file optixpp_namespace.h.

1.2.3.228 void optix::VariableObj::setInt (optix::int2 *i*) [inline, inherited]

Definition at line 3147 of file optixpp_namespace.h.

1.2.3.229 void optix::VariableObj::setInt (int *i1*) [inline, inherited]

Definition at line 3142 of file optixpp_namespace.h.

1.2.3.230 void optix::GeometryObj::setIntersectionProgram (Program *program*) [inline, inherited]

Set the intersection program for this geometry. See rtGeometrySetIntersectionProgram.

Definition at line 2515 of file optixpp_namespace.h.

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

Set the material at given index. See rtGeometryInstanceSetMaterial.

Definition at line 2415 of file optixpp_namespace.h.

1.2.3.232 void optix::GeometryInstanceObj::setMaterialCount (unsigned int *count*) [inline, inherited]

Set the number of materials associated with this instance. See rtGeometryInstanceSetMaterialCount.

Definition at line 2403 of file optixpp_namespace.h.

1.2.3.233 void optix::TransformObj::setMatrix (bool *transpose*, const float * *matrix*, const float * *inverse_matrix*) [inline, inherited]

Set the transform matrix for this node. See rtTransformSetMatrix.

Definition at line 2272 of file optixpp_namespace.h.

1.2.3.234 void optix::VariableObj::setMatrix2x2fv (bool *transpose*, const float * *m*) [inline, inherited]

Definition at line 3041 of file optixpp_namespace.h.

1.2.3.235 void optix::VariableObj::setMatrix2x3fv (bool *transpose*, const float * *m*) [inline, inherited]

Definition at line 3046 of file optixpp_namespace.h.

1.2.3.236 void optix::VariableObj::setMatrix2x4fv (bool *transpose*, const float * *m*) [inline, inherited]

Definition at line 3051 of file optixpp_namespace.h.

1.2.3.237 void optix::VariableObj::setMatrix3x2fv (bool *transpose*, const float * *m*) [inline, inherited]

Definition at line 3056 of file optixpp_namespace.h.

1.2.3.238 void optix::VariableObj::setMatrix3x3fv (bool *transpose*, const float * *m*) [inline, inherited]

Definition at line 3061 of file optixpp_namespace.h.

1.2.3.239 void optix::VariableObj::setMatrix3x4fv (bool *transpose*, const float * *m*) [inline, inherited]

Definition at line 3066 of file optixpp_namespace.h.

1.2.3.240 void optix::VariableObj::setMatrix4x2fv (bool *transpose*, const float * *m*) [inline, inherited]

Definition at line 3071 of file optixpp_namespace.h.

1.2.3.241 void optix::VariableObj::setMatrix4x3fv (bool *transpose*, const float * *m*) [inline, inherited]

Definition at line 3076 of file optixpp_namespace.h.

1.2.3.242 void optix::VariableObj::setMatrix4x4fv (bool *transpose*, const float * *m*) [inline, inherited]

Definition at line 3081 of file optixpp_namespace.h.

1.2.3.243 void optix::TextureSamplerObj::setMaxAnisotropy (float *value*) [inline, inherited]

Set maximum anisotropy for this sampler. See rtTextureSamplerSetMaxAnisotropy.

Definition at line 2719 of file optixpp_namespace.h.

1.2.3.244 void optix::TextureSamplerObj::setMipLevelCount (unsigned int *num_mip_levels*) [inline, inherited]

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

Definition at line 2673 of file optixpp_namespace.h.

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

See rtContextSetMissProgram.

Definition at line 1860 of file optixpp_namespace.h.

1.2.3.246 void optix::GeometryObj::setPrimitiveCount (unsigned int *num_primitives*) [inline, inherited]

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

Definition at line 2491 of file optixpp_namespace.h.

1.2.3.247 void optix::ContextObj::setPrintBufferSize (RTsize *buffer_size_bytes*) [inline, inherited]

See rtContextSetPrintBufferSize.

Definition at line 1912 of file optixpp_namespace.h.

1.2.3.248 void optix::ContextObj::setPrintEnabled (bool *enabled*) [inline, inherited]

See rtContextSetPrintEnabled

Definition at line 1900 of file optixpp_namespace.h.

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

See rtContextSetPrintLaunchIndex.

Definition at line 1924 of file optixpp_namespace.h.

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

Set properties specifying Acceleration builder/traverser behavior. See rtAccelerationSetProperty.

Definition at line 2316 of file optixpp_namespace.h.

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

See rtContextSetRayGenerationProgram

Definition at line 1809 of file optixpp_namespace.h.

1.2.3.252 void optix::ContextObj::setRayTypeCount (unsigned int *num_ray_types*) [inline, inherited]

See rtContextSetRayTypeCount.

Definition at line 1848 of file optixpp_namespace.h.

1.2.3.253 void optix::TextureSamplerObj::setReadMode (RTtexturereadmode *readmode*) [inline, inherited]

Set texture read mode for this sampler. See rtTextureSamplerSetReadMode.

Definition at line 2731 of file optixpp_namespace.h.

1.2.3.254 void optix::BufferObj::setSize (unsigned int *dimensionality*, const RTsize * *dims*) [inline, inherited]

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

Definition at line 2887 of file optixpp_namespace.h.

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

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

Definition at line 2877 of file optixpp_namespace.h.

1.2.3.256 void optix::BufferObj::setSize (RTsize *width*, RTsize *height*) [inline, inherited]

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

Definition at line 2867 of file optixpp_namespace.h.

1.2.3.257 void optix::BufferObj::setSize (RTsize *width*) [inline, inherited]

Set buffer dimensionality to one and buffer width to specified width. See rtBufferSetSize1D.

Definition at line 2857 of file optixpp_namespace.h.

1.2.3.258 void optix::ContextObj::setStackSize (RTsize *stack_size_bytes*) [inline, inherited]

See rtContextSetStackSize

Definition at line 1784 of file optixpp_namespace.h.

1.2.3.259 void optix::VariableObj::setTextureSampler (TextureSampler *texturesample*) [inline, inherited]

Definition at line 3238 of file optixpp_namespace.h.

1.2.3.260 void optix::AccelerationObj::setTraverser (const std::string & *traverser*) [inline, inherited]

Specify the acceleration structure traverser. See rtAccelerationSetTraverser.

Definition at line 2340 of file optixpp_namespace.h.

1.2.3.261 void optix::VariableObj::setUint (unsigned int *u1*, unsigned int *u2*, unsigned int *u3*, unsigned int *u4*) [inline, inherited]

Definition at line 3016 of file optixpp_namespace.h.

1.2.3.262 void optix::VariableObj::setUint (unsigned int *u1*, unsigned int *u2*, unsigned int *u3*) [inline, inherited]

Definition at line 3011 of file optixpp_namespace.h.

1.2.3.263 void optix::VariableObj::setUint (unsigned int *u1*, unsigned int *u2*) [inline, inherited]

Definition at line 3006 of file optixpp_namespace.h.

1.2.3.264 void optix::VariableObj::setUint (unsigned int *u1*) [inline, inherited]

Definition at line 3001 of file optixpp_namespace.h.

1.2.3.265 void optix::VariableObj::setUserData (RTsize *size*, const void * *ptr*) [inline, inherited]

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

Definition at line 3228 of file optixpp_namespace.h.

1.2.3.266 void optix::SelectorObj::setVisitProgram (Program *program*) [inline, inherited]

Set the visitor program for this selector. See rtSelectorSetVisitProgram

Definition at line 2064 of file optixpp_namespace.h.

1.2.3.267 void optix::TextureSamplerObj::setWrapMode (unsigned int *dim*, RTwrapmode *wrapmode*) [inline, inherited]

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

Definition at line 2697 of file optixpp_namespace.h.

1.2.3.268 void optix::BufferObj::unmap () [inline, inherited]

Unmaps a buffer object. See rtBufferUnmap.

Definition at line 2983 of file optixpp_namespace.h.

1.2.3.269 void optix::BufferObj::unregisterGLBuffer () [inline, inherited]

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

Definition at line 2916 of file optixpp_namespace.h.

1.2.3.270 void optix::TextureSamplerObj::unregisterGLTexture () [inline, inherited]

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

Definition at line 2777 of file optixpp_namespace.h.

1.2.3.271 void optix::BufferObj::validate () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2821 of file optixpp_namespace.h.

1.2.3.272 void optix::TextureSamplerObj::validate () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2661 of file optixpp_namespace.h.

1.2.3.273 void optix::MaterialObj::validate () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2582 of file optixpp_namespace.h.

1.2.3.274 void optix::GeometryObj::validate () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2479 of file optixpp_namespace.h.

1.2.3.275 void optix::GeometryInstanceObj::validate () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2379 of file optixpp_namespace.h.

1.2.3.276 void optix::AccelerationObj::validate () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2292 of file optixpp_namespace.h.

1.2.3.277 void optix::TransformObj::validate () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2246 of file optixpp_namespace.h.

1.2.3.278 void optix::GeometryGroupObj::validate () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2188 of file optixpp_namespace.h.

1.2.3.279 void optix::SelectorObj::validate () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2052 of file optixpp_namespace.h.

1.2.3.280 void optix::GroupObj::validate () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 2035 of file optixpp_namespace.h.

1.2.3.281 void optix::ProgramObj::validate () [inline, virtual, inherited]

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

Implements optix::DestroyableObj.

Definition at line 1980 of file optixpp_namespace.h.

1.2.3.282 void optix::ContextObj::validate () [inline, virtual, inherited]

See rtContextValidate.

Implements optix::DestroyableObj.

Definition at line 1493 of file optixpp_namespace.h.

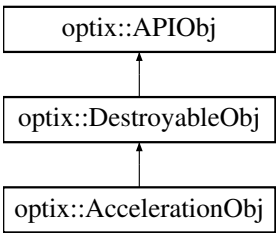
2 Class Documentation

2.1 optix::AccelerationObj Class Reference

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

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::AccelerationObj:



Public Member Functions

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

Friends

- class Handle< AccelerationObj >

- void markDirty ()
- bool isDirty ()

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

- RTsize getDataSize ()
- void getData (void *data)
- 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 1034 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 2287 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 2369 of file optixpp_namespace.h.

2.1.2.3 std::string optix::AccelerationObj::getBuilder () [inline]

Query the acceleration structure builder. See rtAccelerationGetBuilder.

Definition at line 2333 of file optixpp_namespace.h.

2.1.2.4 Context optix::AccelerationObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 2297 of file optixpp_namespace.h.

2.1.2.5 void optix::AccelerationObj::getData (void * *data*) [inline]

Get the marshalled acceleration data. See rtAccelerationGetData.

Definition at line 2359 of file optixpp_namespace.h.

2.1.2.6 RTsize optix::AccelerationObj::getDataSize () [inline]

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

Definition at line 2352 of file optixpp_namespace.h.

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

Query properties specifying Acceleration builder/traverser behavior. See rtAccelerationGetProperty.

Definition at line 2321 of file optixpp_namespace.h.

2.1.2.8 std::string optix::AccelerationObj::getTraverser () [inline]

Query the acceleration structure traverser. See rtAccelerationGetTraverser.

Definition at line 2345 of file optixpp_namespace.h.

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

Query if the acceleration needs a rebuild. See rtAccelerationIsDirty.

Definition at line 2309 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 2304 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 2328 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 rtAccelerationSetData.

Definition at line 2364 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 rtAccelerationSetProperty.

Definition at line 2316 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 2340 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 2292 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 1082 of file optixpp_namespace.h.

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

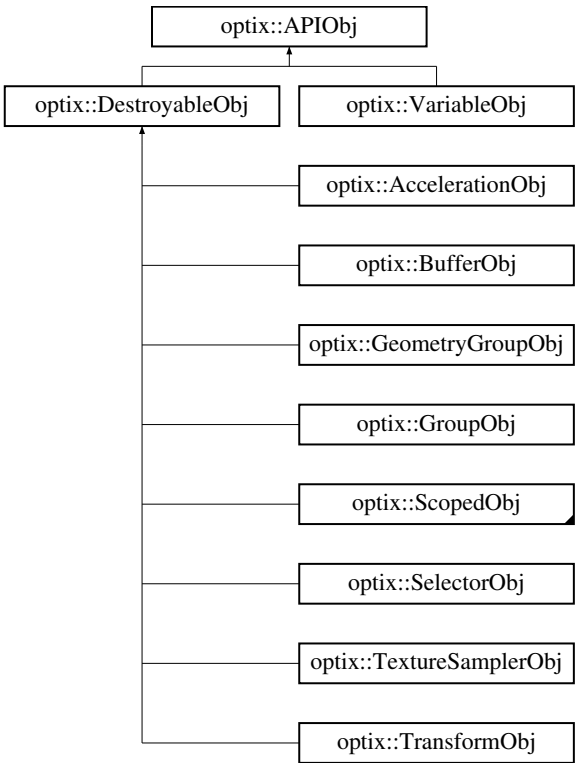
- optixpp_namespace.h

2.2 optix::APIObj Class Reference

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

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::APIObj:



Public Member Functions

- APIObj ()
- virtual ~APIObj ()
- void addReference ()
- int removeReference ()
- virtual Context getContext ()=0
- virtual void checkError (RTresult code)
- void checkErrorNoGetContext (RTresult code)

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 274 of file optixpp_namespace.h.

2.2.2 Constructor & Destructor Documentation

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

Definition at line 276 of file optixpp_namespace.h.

2.2.2.2 virtual optix::APIObj::~APIObj () [inline, virtual]

Definition at line 277 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 280 of file optixpp_namespace.h.

2.2.3.2 void optix::APIObj::checkError (RTresult *code*) [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 1442 of file optixpp_namespace.h.

2.2.3.3 void optix::APIObj::checkErrorNoGetContext (RTresult *code*) [inline]

Definition at line 1450 of file optixpp_namespace.h.

2.2.3.4 virtual Context optix::APIObj::getContext () [pure virtual]

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

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

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

For backwards compatability. Use Exception::makeException instead.

Definition at line 299 of file optixpp_namespace.h.

2.2.3.6 int optix::APIObj::removeReference () [inline]

Decrement the reference count for this object.

Definition at line 282 of file optixpp_namespace.h.

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

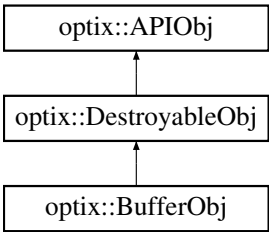
- optixpp_namespace.h

2.3 optix::BufferObj Class Reference

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

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::BufferObj:



Public Member Functions

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

Friends

- class Handle< BufferObj >

- void setFormat (RTformat format)
- RTformat getFormat ()
- void setElementSize (RTsize size_of_element)
- RTsize getElementSize ()
- void setSize (RTsize width)
- void getSize (RTsize &width)
- void setSize (RTsize width, RTsize height)
- void getSize (RTsize &width, RTsize &height)
- void setSize (RTsize width, RTsize height, RTsize depth)
- void getSize (RTsize &width, RTsize &height, RTsize &depth)
- void setSize (unsigned int dimensionality, const RTsize *dims)
- void getSize (unsigned int dimensionality, RTsize *dims)
- unsigned int getDimensionality ()

- unsigned int getGLBOId ()
- 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 1343 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 2816 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 2989 of file optixpp_namespace.h.

2.3.2.3 Context optix::BufferObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 2826 of file optixpp_namespace.h.

2.3.2.4 unsigned int optix::BufferObj::getDimensionality () [inline]

Query dimensionality of buffer. See rtBufferGetDimensionality.

Definition at line 2897 of file optixpp_namespace.h.

2.3.2.5 RTsize optix::BufferObj::getElementSize () [inline]

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

Definition at line 2850 of file optixpp_namespace.h.

2.3.2.6 RTformat optix::BufferObj::getFormat () [inline]

Query the data format for the buffer. See rtBufferGetFormat.

Definition at line 2838 of file optixpp_namespace.h.

2.3.2.7 unsigned int optix::BufferObj::getGLBOId () [inline]

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

Definition at line 2904 of file optixpp_namespace.h.

2.3.2.8 void optix::BufferObj::getSize (unsigned int *dimensionality*, RTsize * *dims*) [inline]

Query dimensions of buffer. See rtBufferGetSizev.

Definition at line 2892 of file optixpp_namespace.h.

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

Query 3D buffer dimension. See rtBufferGetSize3D.

Definition at line 2882 of file optixpp_namespace.h.

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

Query 2D buffer dimension. See rtBufferGetSize2D.

Definition at line 2872 of file optixpp_namespace.h.

2.3.2.11 void optix::BufferObj::getSize (RTsize & *width*) [inline]

Query 1D buffer dimension. See rtBufferGetSize1D.

Definition at line 2862 of file optixpp_namespace.h.

2.3.2.12 void * optix::BufferObj::map () [inline]

Maps a buffer object for host access. See rtBufferMap.

Definition at line 2976 of file optixpp_namespace.h.

2.3.2.13 void optix::BufferObj::registerGLBuffer () [inline]

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

Definition at line 2911 of file optixpp_namespace.h.

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

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

Definition at line 2845 of file optixpp_namespace.h.

2.3.2.15 void optix::BufferObj::setFormat (RTformat *format*) [inline]

Set the data format for the buffer. See rtBufferSetFormat.

Definition at line 2833 of file optixpp_namespace.h.

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

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

Definition at line 2887 of file optixpp_namespace.h.

2.3.2.17 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 2877 of file optixpp_namespace.h.

2.3.2.18 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 2867 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 rtBufferSetSize1D.

Definition at line 2857 of file optixpp_namespace.h.

2.3.2.20 void optix::BufferObj::unmap () [inline]

Unmaps a buffer object. See rtBufferUnmap.

Definition at line 2983 of file optixpp_namespace.h.

2.3.2.21 void optix::BufferObj::unregisterGLBuffer () [inline]

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

Definition at line 2916 of file optixpp_namespace.h.

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

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

Implements optix::DestroyableObj.

Definition at line 2821 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 1435 of file optixpp_namespace.h.

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

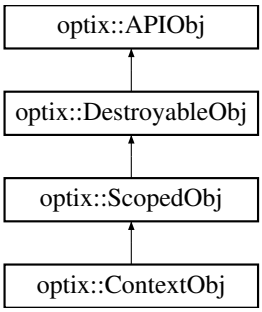
- optixpp_namespace.h

2.4 optix::ContextObj Class Reference

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

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::ContextObj:



Public Member Functions

- void destroy ()
- void validate ()
- Context getContext ()
- void compile ()

- int getRunningState ()
- RTcontext get ()

Static Public Member Functions

- static unsigned int getDeviceCount ()
- static Context create ()

Friends

- class Handle< ContextObj >

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

- 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 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 ()
- unsigned int getEnabledDeviceCount ()

- int getMaxTextureCount ()

- RTsize getAvailableDeviceMemory (int ordinal)

- void setStackSize (RTsize stack_size_bytes)
- RTsize getStackSize ()
- void setEntryPointCount (unsigned int num_entry_points)
- unsigned int getEntryPointCount ()
- void setRayTypeCount (unsigned int num_ray_types)
- unsigned int getRayTypeCount ()

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

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

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

2.4.1 Detailed Description

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

Definition at line 596 of file optixpp_namespace.h.

2.4.2 Member Function Documentation

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

See APIObj::checkError

Reimplemented from optix::APIObj.

Definition at line 1462 of file optixpp_namespace.h.

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

See rtContextCompile.

Definition at line 1872 of file optixpp_namespace.h.

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

Creates a Context object. See rtContextCreate.

Definition at line 1478 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 1498 of file optixpp_namespace.h.

2.4.2.5 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 1541 of file optixpp_namespace.h.

2.4.2.6 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 1532 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 1523 of file optixpp_namespace.h.

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

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

Definition at line 1515 of file optixpp_namespace.h.

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

Create a buffer with given RTbuffertype. See rtBufferCreate.

Definition at line 1508 of file optixpp_namespace.h.

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

Create buffer from GL buffer object. See rtBufferCreateFromGLBO.

Definition at line 1550 of file optixpp_namespace.h.

2.4.2.11 Geometry optix::ContextObj::createGeometry () [inline]

See rtGeometryCreate.

Definition at line 1625 of file optixpp_namespace.h.

2.4.2.12 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 1683 of file optixpp_namespace.h.

2.4.2.13 GeometryGroup optix::ContextObj::createGeometryGroup () [inline]

See rtGeometryGroupCreate.

Definition at line 1675 of file optixpp_namespace.h.

2.4.2.14 template<class Iterator > GeometryInstance optix::ContextObj::createGeometryInstance (Geometry *geometry*, Iterator *matlbegin*, Iterator *matlend*) [inline]

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

Definition at line 1640 of file optixpp_namespace.h.

2.4.2.15 GeometryInstance optix::ContextObj::createGeometryInstance () [inline]

See rtGeometryInstanceCreate.

Definition at line 1632 of file optixpp_namespace.h.

2.4.2.16 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 1662 of file optixpp_namespace.h.

2.4.2.17 Group optix::ContextObj::createGroup () [inline]

See rtGroupCreate.

Definition at line 1654 of file optixpp_namespace.h.

2.4.2.18 Material optix::ContextObj::createMaterial () [inline]

See rtMaterialCreate.

Definition at line 1703 of file optixpp_namespace.h.

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

See rtProgramCreateFromPTXFile.

Definition at line 1710 of file optixpp_namespace.h.

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

See rtProgramCreateFromPTXString.

Definition at line 1717 of file optixpp_namespace.h.

2.4.2.21 Selector optix::ContextObj::createSelector () [inline]

See rtSelectorCreate.

Definition at line 1724 of file optixpp_namespace.h.

2.4.2.22 TextureSampler optix::ContextObj::createTextureSampler () [inline]

See rtTextureSamplerCreate.

Definition at line 1731 of file optixpp_namespace.h.

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

Create TextureSampler from GL image. See rtTextureSamplerCreateFromGLImage.

Definition at line 1618 of file optixpp_namespace.h.

2.4.2.24 Transform optix::ContextObj::createTransform () [inline]

See rtTransformCreate.

Definition at line 1696 of file optixpp_namespace.h.

2.4.2.25 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 1936 of file optixpp_namespace.h.

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

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

Implements optix::DestroyableObj.

Definition at line 1487 of file optixpp_namespace.h.

2.4.2.27 RTcontext optix::ContextObj::get () [inline]

Return the OptiX C API RTcontext object.

Definition at line 1970 of file optixpp_namespace.h.

2.4.2.28 RTsize optix::ContextObj::getAvailableDeviceMemory (int *ordinal*) [inline]

See rtContextGetAttribute.

Definition at line 1775 of file optixpp_namespace.h.

2.4.2.29 Context optix::ContextObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 1457 of file optixpp_namespace.h.

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

Call rtDeviceGetDeviceCount and returns number of valid devices.

Definition at line 1468 of file optixpp_namespace.h.

2.4.2.31 unsigned int optix::ContextObj::getEnabledDeviceCount () [inline]

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

Definition at line 1761 of file optixpp_namespace.h.

2.4.2.32 std::vector< int > optix::ContextObj::getEnabledDevices () [inline]

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

Definition at line 1753 of file optixpp_namespace.h.

2.4.2.33 unsigned int optix::ContextObj::getEntryPointCount () [inline]

See rtContextgetEntryPointCount.

Definition at line 1801 of file optixpp_namespace.h.

2.4.2.34 std::string optix::ContextObj::getErrorString (RTresult *code*) [inline]

See rtContextGetErrroString.

Definition at line 1738 of file optixpp_namespace.h.

2.4.2.35 bool optix::ContextObj::getExceptionEnabled (RTexception *exception*) [inline]

See rtContextGetExceptionEnabled.

Definition at line 1840 of file optixpp_namespace.h.

2.4.2.36 Program optix::ContextObj::getExceptionProgram (unsigned int *entry_point_index*) [inline]

See rtContextGetExceptionProgram.

Definition at line 1827 of file optixpp_namespace.h.

2.4.2.37 int optix::ContextObj::getMaxTextureCount () [inline]

See rtContextGetAttribute

Definition at line 1768 of file optixpp_namespace.h.

2.4.2.38 Program optix::ContextObj::getMissProgram (unsigned int *ray_type_index*) [inline]

See rtContextGetMissProgram.

Definition at line 1865 of file optixpp_namespace.h.

2.4.2.39 RTsize optix::ContextObj::getPrintBufferSize () [inline]

See rtContextGetPrintBufferSize.

Definition at line 1917 of file optixpp_namespace.h.

2.4.2.40 bool optix::ContextObj::getPrintEnabled () [inline]

See rtContextGetPrintEnabled.

Definition at line 1905 of file optixpp_namespace.h.

2.4.2.41 optix::int3 optix::ContextObj::getPrintLaunchIndex () [inline]

See rtContextGetPrintLaunchIndex.

Definition at line 1929 of file optixpp_namespace.h.

2.4.2.42 Program optix::ContextObj::getRayGenerationProgram (unsigned int *entry_point_index*) [inline]

See rtContextGetRayGenerationProgram.

Definition at line 1814 of file optixpp_namespace.h.

2.4.2.43 unsigned int optix::ContextObj::getRayTypeCount () [inline]

See rtContextGetRayTypeCount.

Definition at line 1853 of file optixpp_namespace.h.

2.4.2.44 int optix::ContextObj::getRunningState () [inline]

See rtContextGetRunningState.

Definition at line 1893 of file optixpp_namespace.h.

2.4.2.45 RTsize optix::ContextObj::getStackSize () [inline]

See rtContextGetStackSize.

Definition at line 1789 of file optixpp_namespace.h.

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

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

Implements optix::ScopedObj.

Definition at line 1962 of file optixpp_namespace.h.

2.4.2.47 unsigned int optix::ContextObj::getVariableCount () [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 1955 of file optixpp_namespace.h.

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

See rtContextLaunch3D.

Definition at line 1887 of file optixpp_namespace.h.

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

See rtContextLaunch2D.

Definition at line 1882 of file optixpp_namespace.h.

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

See rtContextLaunch1D

Definition at line 1877 of file optixpp_namespace.h.

2.4.2.51 Variable optix::ContextObj::queryVariable (const std::string & *name*) [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 1943 of file optixpp_namespace.h.

2.4.2.52 void optix::ContextObj::removeVariable (Variable *v*) [inline, virtual]

Remove a variable associated with this object.

Implements optix::ScopedObj.

Definition at line 1950 of file optixpp_namespace.h.

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

See rtContextSetDevices

Definition at line 1746 of file optixpp_namespace.h.

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

See rtContextSetEntryPointCount.

Definition at line 1796 of file optixpp_namespace.h.

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

See rtContextSetExceptionEnabled.

Definition at line 1835 of file optixpp_namespace.h.

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

See rtContextSetExceptionProgram.

Definition at line 1822 of file optixpp_namespace.h.

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

See rtContextSetMissProgram.

Definition at line 1860 of file optixpp_namespace.h.

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

See rtContextSetPrintBufferSize.

Definition at line 1912 of file optixpp_namespace.h.

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

See rtContextSetPrintEnabled

Definition at line 1900 of file optixpp_namespace.h.

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

See rtContextSetPrintLaunchIndex.

Definition at line 1924 of file optixpp_namespace.h.

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

See rtContextSetRayGenerationProgram

Definition at line 1809 of file optixpp_namespace.h.

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

See rtContextSetRayTypeCount.

Definition at line 1848 of file optixpp_namespace.h.

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

See rtContextSetStackSize

Definition at line 1784 of file optixpp_namespace.h.

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

See rtContextValidate.

Implements optix::DestroyableObj.

Definition at line 1493 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 819 of file optixpp_namespace.h.

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

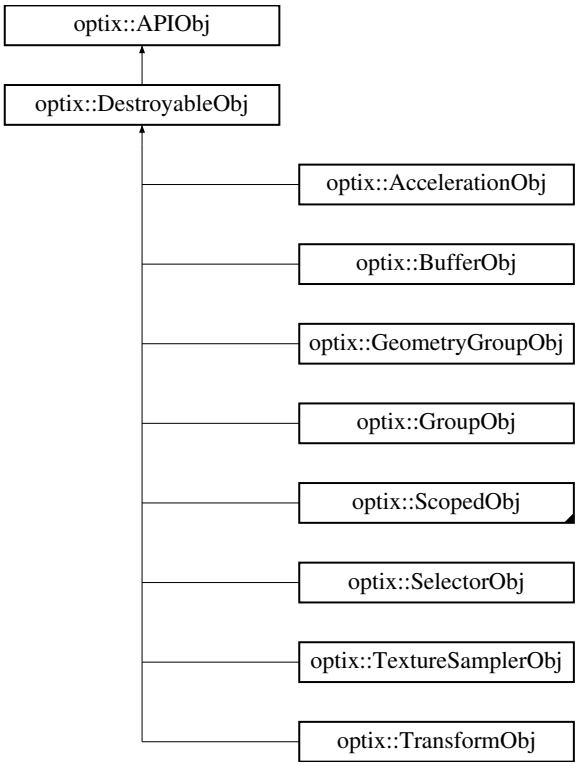
- optixpp_namespace.h

2.5 optix::DestroyableObj Class Reference

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

`#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 323 of file optixpp_namespace.h.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 virtual optix::DestroyableObj::~DestroyableObj () [inline, virtual]

Definition at line 325 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::ContextObj, optix::ProgramObj, optix::GroupObj, optix::GeometryGroupObj, optix::TransformObj, optix::SelectorObj, optix::AccelerationObj, optix::GeometryInstanceObj, optix::GeometryObj, optix::MaterialObj, optix::TextureSamplerObj, and optix::BufferObj.

2.5.3.2 virtual void optix::DestroyableObj::validate () [pure virtual]

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

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

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

- optixpp_namespace.h

2.6 optix::Exception Class Reference

Exception class for error reporting from the OptiXpp API.

`#include <optixpp_namespace.h>`

Public Member Functions

- Exception (const std::string &message, RTresult error_code=RT_ERROR_UNKNOWN)
- 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 218 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 221 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 226 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 232 of file optixpp_namespace.h.

2.6.3.2 const std::string& optix::Exception::getErrorString () const [inline]

Retrieve the error message.

Definition at line 229 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 245 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 239 of file optixpp_namespace.h.

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

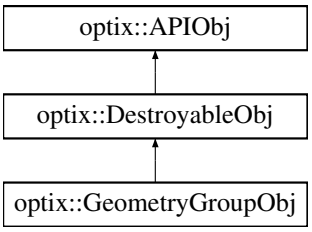
- optixpp_namespace.h

2.7 optix::GeometryGroupObj Class Reference

GeometryGroup wraps the OptiX C API RTgeometrygroup opaque type and its associated function set.

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::GeometryGroupObj:



Public Member Functions

- void destroy ()
- void validate ()
- Context getContext ()
- RTgeometrygroup get ()

Friends

- class Handle< GeometryGroupObj >

- void setAcceleration (Acceleration acceleration)
- Acceleration getAcceleration ()

- void setChildCount (unsigned int count)
- unsigned int getChildCount ()
- void setChild (unsigned int index, GeometryInstance geometryinstance)
- GeometryInstance getChild (unsigned int index)

2.7.1 Detailed Description

GeometryGroup wraps the OptiX C API RTgeometrygroup opaque type and its associated function set.

Definition at line 902 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 2183 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 2236 of file optixpp_namespace.h.

2.7.2.3 Acceleration optix::GeometryGroupObj::getAcceleration () [inline]

Query the Acceleration structure for this group. See rtGeometryGroupGetAcceleration.

Definition at line 2205 of file optixpp_namespace.h.

2.7.2.4 GeometryInstance optix::GeometryGroupObj::getChild (unsigned int *index*) [inline]

Query an indexed GeometryInstance within this group. See rtGeometryGroupGetChild.

Definition at line 2229 of file optixpp_namespace.h.

2.7.2.5 unsigned int optix::GeometryGroupObj::getChildCount () [inline]

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

Definition at line 2217 of file optixpp_namespace.h.

2.7.2.6 Context optix::GeometryGroupObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 2193 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 2200 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 2224 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 2212 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 2188 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 935 of file optixpp_namespace.h.

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

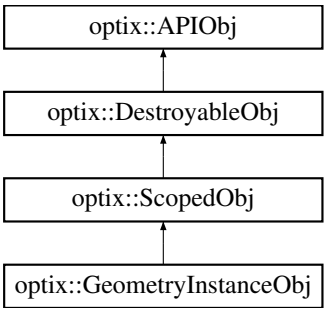
- optixpp_namespace.h

2.8 optix::GeometryInstanceObj Class Reference

GeometryInstance wraps the OptiX C API RTgeometryinstance acceleration opaque type and its associated function set.

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::GeometryInstanceObj:



Public Member Functions

- void destroy ()
- void validate ()
- Context getContext ()
- RTgeometryinstance get ()

Friends

- class Handle< GeometryInstanceObj >

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

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

2.8.1 Detailed Description

GeometryInstance wraps the OptiX C API RTgeometryinstance acceleration opaque type and its associated function set.

Definition at line 1093 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 2428 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 2436 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 2374 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 2469 of file optixpp_namespace.h.

2.8.2.5 Context optix::GeometryInstanceObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 2384 of file optixpp_namespace.h.

2.8.2.6 Geometry optix::GeometryInstanceObj::getGeometry () [inline]

Get the geometry object associated with this instance. See rtGeometryInstanceGetGeometry.

Definition at line 2396 of file optixpp_namespace.h.

2.8.2.7 Material optix::GeometryInstanceObj::getMaterial (unsigned int *idx*) [inline]

Get the material at given index. See rtGeometryInstanceGetMaterial.

Definition at line 2420 of file optixpp_namespace.h.

2.8.2.8 unsigned int optix::GeometryInstanceObj::getMaterialCount () [inline]

Query the number of materials associated with this instance. See rtGeometryInstanceGetMaterialCount.

Definition at line 2408 of file optixpp_namespace.h.

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

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

Implements optix::ScopedObj.

Definition at line 2462 of file optixpp_namespace.h.

2.8.2.10 unsigned int optix::GeometryInstanceObj::getVariableCount () [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 2455 of file optixpp_namespace.h.

2.8.2.11 Variable optix::GeometryInstanceObj::queryVariable (const std::string & *name*) [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 2443 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 2450 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 rtGeometryInstanceSetGeometry.

Definition at line 2391 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 2415 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 rtGeometryInstanceSetMaterialCount.

Definition at line 2403 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 2379 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 1135 of file optixpp_namespace.h.

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

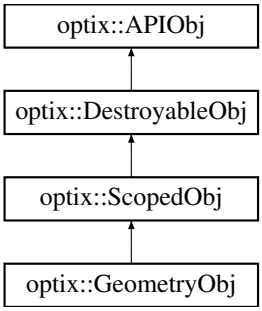
- optixpp_namespace.h

2.9 optix::GeometryObj Class Reference

Geometry wraps the OptiX C API RTgeometry opaque type and its associated function set.

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::GeometryObj:



Public Member Functions

- void destroy ()

- void validate ()
- Context getContext ()
- RTgeometry get ()

Friends

- class Handle< GeometryObj >

- void markDirty ()
- bool isDirty ()

- void setPrimitiveCount (unsigned int num_primitives)
- unsigned int getPrimitiveCount ()

- void setBoundingBoxProgram (Program program)
- Program getBoundingBoxProgram ()
- void setIntersectionProgram (Program program)
- Program getIntersectionProgram ()

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

2.9.1 Detailed Description

Geometry wraps the OptiX C API RTgeometry opaque type and its associated function set.

Definition at line 1145 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 2527 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 2474 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 2572 of file optixpp_namespace.h.

2.9.2.4 Program optix::GeometryObj::getBoundingBoxProgram () [inline]

Get the bounding box program for this geometry. See rtGeometryGetBoundingBoxProgram.

Definition at line 2508 of file optixpp_namespace.h.

2.9.2.5 Context optix::GeometryObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 2484 of file optixpp_namespace.h.

2.9.2.6 Program optix::GeometryObj::getIntersectionProgram () [inline]

Get the intersection program for this geometry. See rtGeometryGetIntersectionProgram.

Definition at line 2520 of file optixpp_namespace.h.

2.9.2.7 unsigned int optix::GeometryObj::getPrimitiveCount () [inline]

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

Definition at line 2496 of file optixpp_namespace.h.

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

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

Implements optix::ScopedObj.

Definition at line 2553 of file optixpp_namespace.h.

2.9.2.9 unsigned int optix::GeometryObj::getVariableCount () [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 2546 of file optixpp_namespace.h.

2.9.2.10 bool optix::GeometryObj::isDirty () [inline]

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

Definition at line 2565 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 2560 of file optixpp_namespace.h.

2.9.2.12 Variable optix::GeometryObj::queryVariable (const std::string & *name*) [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 2534 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 2541 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 2503 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 2515 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 2491 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 2479 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 1195 of file optixpp_namespace.h.

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

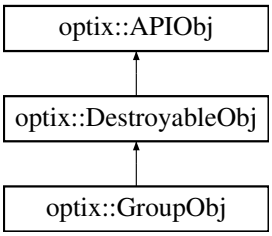
- optixpp_namespace.h

2.10 optix::GroupObj Class Reference

Group wraps the OptiX C API RTgroup opaque type and its associated function set.

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::GroupObj:



Public Member Functions

- void destroy ()
- void validate ()
- Context getContext ()
- RTgroup get ()

Friends

- class Handle< GroupObj >

- void setAcceleration (Acceleration acceleration)

- Acceleration getAcceleration ()

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

2.10.1 Detailed Description

Group wraps the OptiX C API RTgroup opaque type and its associated function set.

Definition at line 858 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 2030 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 2178 of file optixpp_namespace.h.

2.10.2.3 Acceleration optix::GroupObj::getAcceleration () [inline]

Query the Acceleration structure for this group. See rtGroupGetAcceleration.

Definition at line 2145 of file optixpp_namespace.h.

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

Query an indexed child within this group. See rtGroupGetChild.

Definition at line 2171 of file optixpp_namespace.h.

2.10.2.5 unsigned int optix::GroupObj::getChildCount () [inline]

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

Definition at line 2157 of file optixpp_namespace.h.

2.10.2.6 Context optix::GroupObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 2040 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 2140 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 2165 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 2152 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 2035 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 892 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

The Handle class is a reference counted handle class used to manipulate API objects.

`#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-> ()
- T * get ()
- 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 108 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 111 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 114 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 118 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 121 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 125 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 137 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 180 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 149 of file optixpp_namespace.h.

2.11.3.3 template<class T> static unsigned int optix::Handle< T >::getDeviceCount () [inline, static]

Query the machine device count. Only valid for contexts.

Definition at line 183 of file optixpp_namespace.h.

2.11.3.4 template<class T> optix::Handle< T >::operator bool () const [inline]

implicit bool cast based on NULLness of wrapped pointer

Definition at line 152 of file optixpp_namespace.h.

2.11.3.5 template<class T> T* optix::Handle< T >::operator-> () [inline]

Dereferences the handle.

Definition at line 146 of file optixpp_namespace.h.

2.11.3.6 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 133 of file optixpp_namespace.h.

2.11.3.7 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 128 of file optixpp_namespace.h.

2.11.3.8 template<class T > Handle< VariableObj > optix::Handle< T >::operator[ ] (const char * *varname*) [inline]

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 584 of file optixpp_namespace.h.

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

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 575 of file optixpp_namespace.h.

2.11.3.10 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 143 of file optixpp_namespace.h.

2.11.3.11 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 140 of file optixpp_namespace.h.

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

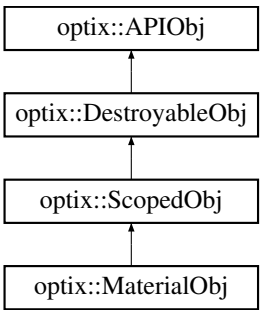
- optixpp_namespace.h

2.12 optix::MaterialObj Class Reference

Material wraps the OptiX C API RTmaterial opaque type and its associated function set.

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::MaterialObj:



Public Member Functions

- void destroy ()
- void validate ()
- Context getContext ()
- RTmaterial get ()

Friends

- class Handle< MaterialObj >

- void setClosestHitProgram (unsigned int ray_type_index, Program program)
- Program getClosestHitProgram (unsigned int ray_type_index)
- void setAnyHitProgram (unsigned int ray_type_index, Program program)
- Program getAnyHitProgram (unsigned int ray_type_index)

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

2.12.1 Detailed Description

Material wraps the OptiX C API RTmaterial opaque type and its associated function set.

Definition at line 1205 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 2618 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 2577 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 2651 of file optixpp_namespace.h.

2.12.2.4 Program optix::MaterialObj::getAnyHitProgram (unsigned int *ray_type_index*) [inline]

Get any hit program for this material at the given *ray_type* index. See rtMaterialGetAnyHitProgram.

Definition at line 2611 of file optixpp_namespace.h.

2.12.2.5 Program optix::MaterialObj::getClosestHitProgram (unsigned int *ray_type_index*) [inline]

Get closest hit program for this material at the given *ray_type* index. See rtMaterialGetClosestHitProgram.

Definition at line 2599 of file optixpp_namespace.h.

2.12.2.6 Context optix::MaterialObj::getContext () [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.

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

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

Implements optix::ScopedObj.

Definition at line 2644 of file optixpp_namespace.h.

2.12.2.8 unsigned int optix::MaterialObj::getVariableCount () [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 2637 of file optixpp_namespace.h.

2.12.2.9 Variable optix::MaterialObj::queryVariable (const std::string & *name*) [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 2625 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 2632 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 rtMaterialSetAnyHitProgram.

Definition at line 2606 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 rtMaterialSetClosestHitProgram.

Definition at line 2594 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 2582 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 1238 of file optixpp_namespace.h.

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

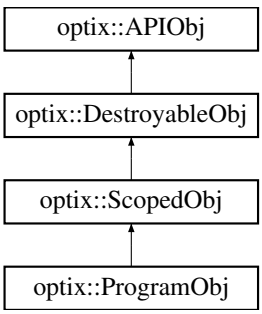
- optixpp_namespace.h

2.13 optix::ProgramObj Class Reference

Program object wraps the OptiX C API RTprogram opaque type and its associated function set.

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::ProgramObj:



Public Member Functions

- void destroy ()
- void validate ()
- Context getContext ()
- Variable declareVariable (const std::string &name)

- Variable queryVariable (const std::string &name)
- void removeVariable (Variable v)
- unsigned int getVariableCount ()
- Variable getVariable (unsigned int index)
- 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 829 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 1992 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 1975 of file optixpp_namespace.h.

2.13.2.3 RTprogram optix::ProgramObj::get () [inline]

Definition at line 2025 of file optixpp_namespace.h.

2.13.2.4 Context optix::ProgramObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 1985 of file optixpp_namespace.h.

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

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

Implements optix::ScopedObj.

Definition at line 2018 of file optixpp_namespace.h.

2.13.2.6 unsigned int optix::ProgramObj::getVariableCount () [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 2011 of file optixpp_namespace.h.

2.13.2.7 Variable optix::ProgramObj::queryVariable (const std::string & *name*) [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 1999 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 2006 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 1980 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 848 of file optixpp_namespace.h.

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

- optixpp_namespace.h

2.14 RTUtraversalresult Struct Reference

Structure encapsulating the result of a single ray query.

`#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:

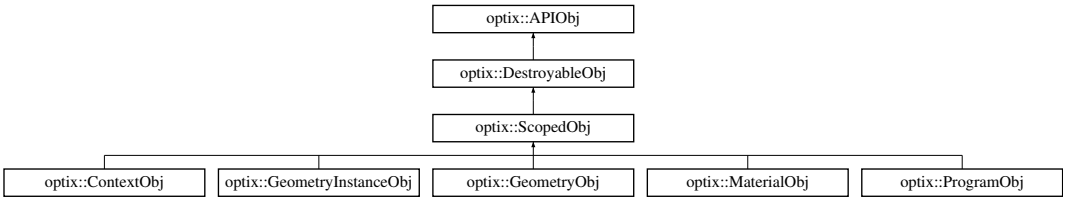
- optixu_traversal.h

2.15 optix::ScopedObj Class Reference

Base class for all objects which are OptiX variable containers.

`#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)=0
- virtual void removeVariable (Variable v)=0
- virtual unsigned int getVariableCount ()=0
- virtual Variable getVariable (unsigned int index)=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 349 of file optixpp_namespace.h.

2.15.2 Constructor & Destructor Documentation

2.15.2.1 virtual optix::ScopedObj::~ScopedObj () [inline, virtual]

Definition at line 351 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::ContextObj, optix::ProgramObj, optix::GeometryInstanceObj, optix::GeometryObj, and optix::MaterialObj.

2.15.3.2 virtual Variable optix::ScopedObj::getVariable (unsigned int *index*) [pure virtual]

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

Implemented in optix::ContextObj, optix::ProgramObj, optix::GeometryInstanceObj, optix::GeometryObj, and optix::MaterialObj.

2.15.3.3 virtual unsigned int optix::ScopedObj::getVariableCount () [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::ContextObj, optix::ProgramObj, optix::GeometryInstanceObj, optix::GeometryObj, and optix::MaterialObj.

2.15.3.4 virtual Variable optix::ScopedObj::queryVariable (const std::string & *name*) [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::ContextObj, optix::ProgramObj, optix::GeometryInstanceObj, optix::GeometryObj, and optix::MaterialObj.

2.15.3.5 virtual void optix::ScopedObj::removeVariable (Variable *v*) [pure virtual]

Remove a variable associated with this object.

Implemented in optix::ContextObj, optix::ProgramObj, optix::GeometryInstanceObj, optix::GeometryObj, and optix::MaterialObj.

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

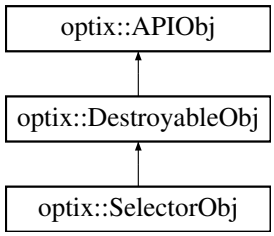
- optixpp_namespace.h

2.16 optix::SelectorObj Class Reference

Selector wraps the OptiX C API RTselector opaque type and its associated function set.

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::SelectorObj:



Public Member Functions

- void destroy ()
- void validate ()
- Context getContext ()
- RTselector get ()

Friends

- class Handle< SelectorObj >

- void setVisitProgram (Program program)
- Program getVisitProgram ()

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

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

2.16.1 Detailed Description

Selector wraps the OptiX C API RTselector opaque type and its associated function set.

Definition at line 983 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 2102 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 2047 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 2135 of file optixpp_namespace.h.

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

Query an indexed child within this group. See rtSelectorGetChild.

Definition at line 2095 of file optixpp_namespace.h.

2.16.2.5 unsigned int optix::SelectorObj::getChildCount () [inline]

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

Definition at line 2081 of file optixpp_namespace.h.

2.16.2.6 Context optix::SelectorObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 2057 of file optixpp_namespace.h.

2.16.2.7 Variable optix::SelectorObj::getVariable (unsigned int *index*) [inline]

Definition at line 2128 of file optixpp_namespace.h.

2.16.2.8 unsigned int optix::SelectorObj::getVariableCount () [inline]

Definition at line 2121 of file optixpp_namespace.h.

2.16.2.9 Program optix::SelectorObj::getVisitProgram () [inline]

Get the visitor program for this selector. See rtSelectorGetVisitProgram.

Definition at line 2069 of file optixpp_namespace.h.

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

Definition at line 2109 of file optixpp_namespace.h.

2.16.2.11 void optix::SelectorObj::removeVariable (Variable *v*) [inline]

Definition at line 2116 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 2089 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 2076 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 2064 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 2052 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 1024 of file optixpp_namespace.h.

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

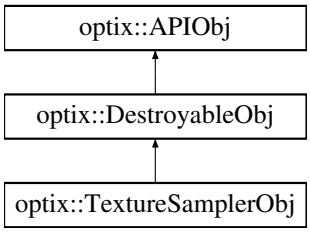
- optixpp_namespace.h

2.17 optix::TextureSamplerObj Class Reference

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

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::TextureSamplerObj:



Public Member Functions

- void destroy ()
- void validate ()
- Context getContext ()
- RTtexturesampler get ()

Friends

- class Handle< TextureSamplerObj >

- void setMipLevelCount (unsigned int num_mip_levels)
- unsigned int getMipLevelCount ()
- void setArraySize (unsigned int num_textures_in_array)
- unsigned int getArraySize ()
- void setWrapMode (unsigned int dim, RTwrapmode wrapmode)
- RTwrapmode getWrapMode (unsigned int dim)
- void setFilteringModes (RTfiltermode minification, RTfiltermode magnification, RTfiltermode mipmapping)
- void getFilteringModes (RTfiltermode &minification, RTfiltermode &magnification, RTfiltermode &mipmapping)
- void setMaxAnisotropy (float value)
- float getMaxAnisotropy ()
- void setReadMode (RTtexturereadmode readmode)
- RTtexturereadmode getReadMode ()
- void setIndexingMode (RTtextureindexmode indexmode)
- RTtextureindexmode getIndexingMode ()

- void setBuffer (unsigned int texture_array_idx, unsigned int mip_level, Buffer buffer)
- Buffer getBuffer (unsigned int texture_array_idx, unsigned int mip_level)

- 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 1248 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 2656 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 2767 of file optixpp_namespace.h.

2.17.2.3 unsigned int optix::TextureSamplerObj::getArraySize () [inline]

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

Definition at line 2690 of file optixpp_namespace.h.

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

Get the underlying buffer used for texture storage. rtTextureSamplerGetBuffer.

Definition at line 2760 of file optixpp_namespace.h.

2.17.2.5 Context optix::TextureSamplerObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 2666 of file optixpp_namespace.h.

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

Query filtering modes for this sampler. See rtTextureSamplerGetFilteringModes.

Definition at line 2714 of file optixpp_namespace.h.

2.17.2.7 RTtextureindexmode optix::TextureSamplerObj::getIndexingMode () [inline]

Query texture indexing mode for this sampler. See rtTextureSamplerGetIndexingMode.

Definition at line 2748 of file optixpp_namespace.h.

2.17.2.8 float optix::TextureSamplerObj::getMaxAnisotropy () [inline]

Query maximum anisotropy for this sampler. See rtTextureSamplerGetMaxAnisotropy.

Definition at line 2724 of file optixpp_namespace.h.

2.17.2.9 unsigned int optix::TextureSamplerObj::getMipLevelCount () [inline]

Query the number of mip levels for this sampler. See rtTextureSamplerGetMipLevelCount.

Definition at line 2678 of file optixpp_namespace.h.

2.17.2.10 RTtexturereadmode optix::TextureSamplerObj::getReadMode () [inline]

Query texture read mode for this sampler. See rtTextureSamplerGetReadMode.

Definition at line 2736 of file optixpp_namespace.h.

2.17.2.11 RTwrapmode optix::TextureSamplerObj::getWrapMode (unsigned int *dim*) [inline]

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

Definition at line 2702 of file optixpp_namespace.h.

2.17.2.12 void optix::TextureSamplerObj::registerGLTexture () [inline]

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

Definition at line 2772 of file optixpp_namespace.h.

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

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

Definition at line 2685 of file optixpp_namespace.h.

2.17.2.14 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 2755 of file optixpp_namespace.h.

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

Set filtering modes for this sampler. See rtTextureSamplerSetFilteringModes.

Definition at line 2709 of file optixpp_namespace.h.

2.17.2.16 void optix::TextureSamplerObj::setIndexingMode (RTtextureindexmode *indexmode*) [inline]

Set texture indexing mode for this sampler. See rtTextureSamplerSetIndexingMode.

Definition at line 2743 of file optixpp_namespace.h.

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

Set maximum anisotropy for this sampler. See rtTextureSamplerSetMaxAnisotropy.

Definition at line 2719 of file optixpp_namespace.h.

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

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

Definition at line 2673 of file optixpp_namespace.h.

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

Set texture read mode for this sampler. See rtTextureSamplerSetReadMode.

Definition at line 2731 of file optixpp_namespace.h.

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

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

Definition at line 2697 of file optixpp_namespace.h.

2.17.2.21 void optix::TextureSamplerObj::unregisterGLTexture () [inline]

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

Definition at line 2777 of file optixpp_namespace.h.

2.17.2.22 void optix::TextureSamplerObj::validate () [inline, virtual]

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

Implements optix::DestroyableObj.

Definition at line 2661 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 1333 of file optixpp_namespace.h.

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

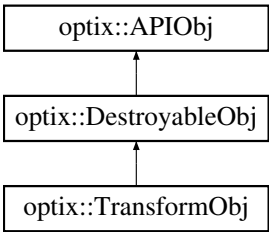
- optixpp_namespace.h

2.18 optix::TransformObj Class Reference

Transform wraps the OptiX C API RTtransform opaque type and its associated function set.

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::TransformObj:



Public Member Functions

- void destroy ()
- void validate ()
- Context getContext ()
- RTtransform get ()

Friends

- class Handle< TransformObj >

- template<typename T >
  void setChild (T child)
- template<typename T >
  T getChild ()

- void setMatrix (bool transpose, const float *matrix, const float *inverse_matrix)
- void getMatrix (bool transpose, float *matrix, float *inverse_matrix)

2.18.1 Detailed Description

Transform wraps the OptiX C API RTtransform opaque type and its associated function set.

Definition at line 945 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 2241 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 2282 of file optixpp_namespace.h.

2.18.2.3 template<typename T > T optix::TransformObj::getChild () [inline]

Set the child node of this transform. See rtTransformGetChild.

Definition at line 2265 of file optixpp_namespace.h.

2.18.2.4 Context optix::TransformObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 2251 of file optixpp_namespace.h.

2.18.2.5 void optix::TransformObj::getMatrix (bool *transpose*, float * *matrix*, float * *inverse_matrix*) [inline]

Get the transform matrix for this node. See rtTransformGetMatrix.

Definition at line 2277 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 2259 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 2272 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 2246 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 973 of file optixpp_namespace.h.

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

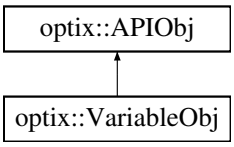
- optixpp_namespace.h

2.19 optix::VariableObj Class Reference

Variable object wraps OptiX C API RTvariable type and its related function set.

`#include <optixpp_namespace.h>`

Inheritance diagram for optix::VariableObj:



Public Member Functions

- Context getContext ()
- std::string getName ()
- std::string getAnnotation ()
- RTobjecttype getType ()
- RTvariable get ()
- RTsize getSize ()

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 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 ()
- unsigned int getUint ()
- int getInt ()

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 (Selector selector)
- void set (Transform transform)

OptiX API object getters

Reitrieve OptiX API object value from a variable

- Buffer getBuffer ()
- TextureSampler getTextureSampler ()

User data variable accessors

- void setUserData (RTsize size, const void *ptr)
- void getUserData (RTsize size, void *ptr)

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 388 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 3297 of file optixpp_namespace.h.

2.19.2.2 std::string optix::VariableObj::getAnnotation () [inline]

Retrieve the annotation associated with the variable.

Definition at line 3283 of file optixpp_namespace.h.

2.19.2.3 Buffer optix::VariableObj::getBuffer () [inline]

Definition at line 3268 of file optixpp_namespace.h.

2.19.2.4 Context optix::VariableObj::getContext () [inline, virtual]

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

Implements optix::APIObj.

Definition at line 2994 of file optixpp_namespace.h.

2.19.2.5 float optix::VariableObj::getFloat () [inline]

Definition at line 3197 of file optixpp_namespace.h.

2.19.2.6 int optix::VariableObj::getInt () [inline]

Definition at line 3211 of file optixpp_namespace.h.

2.19.2.7 std::string optix::VariableObj::getName () [inline]

Retrieve the name of the variable.

Definition at line 3276 of file optixpp_namespace.h.

2.19.2.8 RTsize optix::VariableObj::getSize () [inline]

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

Definition at line 3302 of file optixpp_namespace.h.

2.19.2.9 optix::TextureSampler optix::VariableObj::getTextureSampler () [inline]

Definition at line 3309 of file optixpp_namespace.h.

2.19.2.10 RTobjecttype optix::VariableObj::getType () [inline]

Query the object type of the variable.

Definition at line 3290 of file optixpp_namespace.h.

2.19.2.11 unsigned int optix::VariableObj::getUint () [inline]

Definition at line 3204 of file optixpp_namespace.h.

2.19.2.12 void optix::VariableObj::getUserData (RTsize *size*, void * *ptr*) [inline]

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

Definition at line 3233 of file optixpp_namespace.h.

2.19.2.13 void optix::VariableObj::set (Transform *transform*)

2.19.2.14 void optix::VariableObj::set (Selector *selector*)

2.19.2.15 void optix::VariableObj::set (Group *group*)

2.19.2.16 void optix::VariableObj::set (GeometryGroup *group*)

2.19.2.17 void optix::VariableObj::set (TextureSampler *texturesample*)

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

Definition at line 3223 of file optixpp_namespace.h.

2.19.2.19 void optix::VariableObj::set1fv (const float * *f*) [inline]

Set variable value to a scalar float.

Definition at line 3121 of file optixpp_namespace.h.

2.19.2.20 void optix::VariableObj::set1iv (const int * *i*) [inline]

Definition at line 3177 of file optixpp_namespace.h.

2.19.2.21 void optix::VariableObj::set1uiv (const unsigned int * u) [inline]

Definition at line 3021 of file optixpp_namespace.h.

2.19.2.22 void optix::VariableObj::set2fv (const float * f) [inline]

Set variable value to a float2.

Definition at line 3126 of file optixpp_namespace.h.

2.19.2.23 void optix::VariableObj::set2iv (const int * i) [inline]

Definition at line 3182 of file optixpp_namespace.h.

2.19.2.24 void optix::VariableObj::set2uiv (const unsigned int * u) [inline]

Definition at line 3026 of file optixpp_namespace.h.

2.19.2.25 void optix::VariableObj::set3fv (const float * f) [inline]

Set variable value to a float3.

Definition at line 3131 of file optixpp_namespace.h.

2.19.2.26 void optix::VariableObj::set3iv (const int * i) [inline]

Definition at line 3187 of file optixpp_namespace.h.

2.19.2.27 void optix::VariableObj::set3uiv (const unsigned int * u) [inline]

Definition at line 3031 of file optixpp_namespace.h.

2.19.2.28 void optix::VariableObj::set4fv (const float * f) [inline]

Set variable value to a float4.

Definition at line 3136 of file optixpp_namespace.h.

2.19.2.29 void optix::VariableObj::set4iv (const int * *i*) [inline]

Definition at line 3192 of file optixpp_namespace.h.

2.19.2.30 void optix::VariableObj::set4uiv (const unsigned int * *u*) [inline]

Definition at line 3036 of file optixpp_namespace.h.

2.19.2.31 void optix::VariableObj::setBuffer (Buffer *buffer*) [inline]

Definition at line 3218 of file optixpp_namespace.h.

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

Set variable value to a float4.

Definition at line 3116 of file optixpp_namespace.h.

2.19.2.33 void optix::VariableObj::setFloat (optix::float4 *f*) [inline]

Set variable value to a float4.

Definition at line 3111 of file optixpp_namespace.h.

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

Set variable value to a float3.

Definition at line 3106 of file optixpp_namespace.h.

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

Set variable value to a float3.

Definition at line 3101 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 3096 of file optixpp_namespace.h.

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

Set variable value to a float2.

Definition at line 3091 of file optixpp_namespace.h.

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

Set variable value to a scalar float.

Definition at line 3086 of file optixpp_namespace.h.

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

Definition at line 3167 of file optixpp_namespace.h.

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

Definition at line 3172 of file optixpp_namespace.h.

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

Definition at line 3157 of file optixpp_namespace.h.

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

Definition at line 3162 of file optixpp_namespace.h.

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

Definition at line 3147 of file optixpp_namespace.h.

2.19.2.44 void optix::VariableObj::setInt (int *i1*, int *i2*) [inline]

Definition at line 3152 of file optixpp_namespace.h.

2.19.2.45 void optix::VariableObj::setInt (int *i1*) [inline]

Definition at line 3142 of file optixpp_namespace.h.

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

Definition at line 3041 of file optixpp_namespace.h.

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

Definition at line 3046 of file optixpp_namespace.h.

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

Definition at line 3051 of file optixpp_namespace.h.

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

Definition at line 3056 of file optixpp_namespace.h.

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

Definition at line 3061 of file optixpp_namespace.h.

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

Definition at line 3066 of file optixpp_namespace.h.

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

Definition at line 3071 of file optixpp_namespace.h.

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

Definition at line 3076 of file optixpp_namespace.h.

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

Definition at line 3081 of file optixpp_namespace.h.

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

Definition at line 3238 of file optixpp_namespace.h.

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

Definition at line 3016 of file optixpp_namespace.h.

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

Definition at line 3011 of file optixpp_namespace.h.

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

Definition at line 3006 of file optixpp_namespace.h.

2.19.2.59 void optix::VariableObj::setUint (unsigned int *u1*) [inline]

Definition at line 3001 of file optixpp_namespace.h.

2.19.2.60 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 3228 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 570 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

A C++ wrapper around the OptiX API.

```
#include <optix.h>
#include <optix_gl_interop.h>
#include <string>
#include <vector>
#include "optixu_vector_types.h"
#include <limits.h>
#include <stddef.h>
#include <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

```
00001
00002 /*
00003  * Copyright (c) 2008 - 2009 NVIDIA Corporation.  All rights reserved.
00004  *
00005  * NVIDIA Corporation and its licensors retain all intellectual property and prop
      rietary
00006  * rights in and to this software, related documentation and any modifications th
      ereto.
00007  * Any use, reproduction, disclosure or distribution of this software and related

00008  * documentation without an express license agreement from NVIDIA Corporation is
      strictly
00009  * prohibited.
00010  *
00011  * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *
      AS IS*
00012  * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIE
      D,
00013  * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNE
      SS FOR A
00014  * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR A
      NY
00015  * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING,
       WITHOUT
00016  * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS
      OF
00017  * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF O
      R
00018  * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBI
      LITY OF
00019  * SUCH DAMAGES
00020  */
00021
00022
00052
00057
00058
00059 #ifndef __optixu_optixpp_namespace_h__
00060 #define __optixu_optixpp_namespace_h__
00061
00062 #include <optix.h>
00063
00064 #ifdef _WIN32
00065 #  ifndef WIN32_LEAN_AND_MEAN
00066 #    define WIN32_LEAN_AND_MEAN
00067 #  endif
00068 #  include<windows.h>
00069 #  include<optix_d3d9_interop.h>
00070 #  include<optix_d3d10_interop.h>
00071 #  include<optix_d3d11_interop.h>
00072 #endif
00073 #include <optix_gl_interop.h>
00074
00075 #include <string>
00076 #include <vector>
00077 #include "optixu_vector_types.h"
00078
00079 //-----------------------------------------------------------------------------
00080 //
00081 // Doxygen group specifications
00082 //
00083 //-----------------------------------------------------------------------------
00084
00085 //-----------------------------------------------------------------------------
00086 //
```

```
00087 // C++ API
00088 //
00089 //-----------------------------------------------------------------------------
00090
00091 namespace optix {
00094
00095   class AccelerationObj;
00096   class BufferObj;
00097   class ContextObj;
00098   class GeometryObj;
00099   class GeometryGroupObj;
00100   class GeometryInstanceObj;
00101   class GroupObj;
00102   class MaterialObj;
00103   class ProgramObj;
00104   class SelectorObj;
00105   class TextureSamplerObj;
00106   class TransformObj;
00107   class VariableObj;
00108
00109   class APIObj;
00110   class ScopedObj;
00111
00112
00120   template<class T>
00121   class Handle {
00122   public:
00124     Handle() : ptr(0) {}
00125
00127     Handle(T* ptr) : ptr(ptr) { ref(); }
00128
00130     template<class U>
00131     Handle(U* ptr) : ptr(ptr) { ref(); }
00132
00134     Handle(const Handle<T>& copy) : ptr(copy.ptr) { ref(); }
00135
00137     template<class U>
00138     Handle(const Handle<U>& copy) : ptr(copy.ptr) { ref(); }
00139
00141     Handle<T>& operator=(const Handle<T>& copy)
00142     { if(ptr != copy.ptr) { unref(); ptr = copy.ptr; ref(); } return *this; }
00143
00145     template<class U>
00146     Handle<T>& operator=( const Handle<U>& copy)
00147     { if(ptr != copy.ptr) { unref(); ptr = copy.ptr; ref(); } return *this; }
00148
00150     ~Handle() { unref(); }
00151
00153     static Handle<T> take( typename T::api_t p ) { return p? new T(p) : 0; }
00156     static Handle<T> take( RTobject p ) { return p? new T(static_cast<typename T:
      :api_t>(p)) : 0; }
00157
00159     T* operator->() { return ptr; }
00160
00162     T* get() { return ptr; }
00163
00165     operator bool() const  { return ptr != 0; }
00166
00170     Handle<VariableObj> operator[](const std::string& varname);
00171
00190     Handle<VariableObj> operator[](const char* varname);
00191
00193     static Handle<T> create() { return T::create(); }
00194
00196     static unsigned int getDeviceCount() { return T::getDeviceCount(); }
00197
00198   private:
```

```
00199     inline void ref() { if(ptr) ptr->addReference(); }
00200     inline void unref() { if(ptr && ptr->removeReference() == 0) delete ptr; }
00201     T* ptr;
00202   };
00203
00204
00205   //------------------------------------------------------------------------
00206
00207   typedef Handle<AccelerationObj>     Acceleration;
00208   typedef Handle<BufferObj>           Buffer;
00209   typedef Handle<ContextObj>          Context;
00210   typedef Handle<GeometryObj>         Geometry;
00211   typedef Handle<GeometryGroupObj>    GeometryGroup;
00212   typedef Handle<GeometryInstanceObj> GeometryInstance;
00213   typedef Handle<GroupObj>            Group;
00214   typedef Handle<MaterialObj>         Material;
00215   typedef Handle<ProgramObj>          Program;
00216   typedef Handle<SelectorObj>         Selector;
00217   typedef Handle<TextureSamplerObj>   TextureSampler;
00218   typedef Handle<TransformObj>        Transform;
00219   typedef Handle<VariableObj>         Variable;
00220
00221
00222   //------------------------------------------------------------------------
00223
00224
00231   class Exception: public std::exception {
00232   public:
00234     Exception( const std::string& message, RTresult error_code = RT_ERROR_UNKNOWN
      )
00235       : m_message(message), m_error_code( error_code ) {}
00236
00239     virtual ~Exception() throw() {}
00240
00242     const std::string& getErrorString() const { return m_message; }
00243
00245     RTresult getErrorCode() const { return m_error_code; }
00246
00249     static Exception makeException( RTresult code, RTcontext context );
00250
00252     virtual const char* what() const throw() { return getErrorString().c_str(); }

00253   private:
00254     std::string m_message;
00255     RTresult    m_error_code;
00256   };
00257
00258   inline Exception Exception::makeException( RTresult code, RTcontext context )
00259   {
00260     const char* str;
00261     rtContextGetErrorString( context, code, &str);
00262     return Exception( std::string(str), code );
00263   }
00264
00265
00266   //------------------------------------------------------------------------
00267
00268
00287   class APIObj {
00288   public:
00289     APIObj() : ref_count(0) {}
00290     virtual ~APIObj() {}
00291
00293     void addReference()    { ++ref_count; }
00295     int  removeReference() { return --ref_count; }
00296
00298     virtual Context getContext()=0;
```

```
00299
00302     virtual void checkError(RTresult code);
00303
00304     void checkErrorNoGetContext(RTresult code);
00305
00307     static Exception makeException( RTresult code, RTcontext context );
00308   private:
00309     int ref_count;
00310   };
00311
00312   inline Exception APIObj::makeException( RTresult code, RTcontext context )
00313   {
00314     return Exception::makeException( code, context );
00315   }
00316
00317
00318   //------------------------------------------------------------------------
00319
00320
00336   class DestroyableObj : public APIObj {
00337   public:
00338     virtual ~DestroyableObj() {}
00339
00341     virtual void destroy() = 0;
00342
00344     virtual void validate() = 0;
00345   };
00346
00347
00348
00349   //------------------------------------------------------------------------
00350
00351
00362   class ScopedObj : public DestroyableObj {
00363   public:
00364     virtual ~ScopedObj() {}
00365
00368     virtual Variable declareVariable (const std::string& name) = 0;
00371     virtual Variable queryVariable   (const std::string& name) = 0;
00373     virtual void     removeVariable  (Variable v) = 0;
00377     virtual unsigned int getVariableCount() = 0;
00379     virtual Variable getVariable     (unsigned int index) = 0;
00380   };
00381
00382
00383
00384   //------------------------------------------------------------------------
00385
00386
00401   class VariableObj : public APIObj {
00402   public:
00403
00404     Context getContext();
00405
00408
00409
00410     void setFloat(float f1);
00412     void setFloat(optix::float2 f);
00414     void setFloat(float f1, float f2);
00416     void setFloat(optix::float3 f);
00418     void setFloat(float f1, float f2, float f3);
00420     void setFloat(optix::float4 f);
00422     void setFloat(float f1, float f2, float f3, float f4);
00424     void set1fv(const float* f);
00426     void set2fv(const float* f);
00428     void set3fv(const float* f);
00430     void set4fv(const float* f);
```

```
00432
00435
00436     void setInt(int i1);
00437     void setInt(int i1, int i2);
00438     void setInt(optix::int2 i);
00439     void setInt(int i1, int i2, int i3);
00440     void setInt(optix::int3 i);
00441     void setInt(int i1, int i2, int i3, int i4);
00442     void setInt(optix::int4 i);
00443     void set1iv(const int* i);
00444     void set2iv(const int* i);
00445     void set3iv(const int* i);
00446     void set4iv(const int* i);
00448
00451
00452     void setUint(unsigned int u1);
00453     void setUint(unsigned int u1, unsigned int u2);
00454     void setUint(unsigned int u1, unsigned int u2, unsigned int u3);
00455     void setUint(unsigned int u1, unsigned int u2, unsigned int u3, unsigned int
      u4);
00456     void set1uiv(const unsigned int* u);
00457     void set2uiv(const unsigned int* u);
00458     void set3uiv(const unsigned int* u);
00459     void set4uiv(const unsigned int* u);
00461
00464
00465     void setMatrix2x2fv(bool transpose, const float* m);
00466     void setMatrix2x3fv(bool transpose, const float* m);
00467     void setMatrix2x4fv(bool transpose, const float* m);
00468     void setMatrix3x2fv(bool transpose, const float* m);
00469     void setMatrix3x3fv(bool transpose, const float* m);
00470     void setMatrix3x4fv(bool transpose, const float* m);
00471     void setMatrix4x2fv(bool transpose, const float* m);
00472     void setMatrix4x3fv(bool transpose, const float* m);
00473     void setMatrix4x4fv(bool transpose, const float* m);
00475
00478
00479     float getFloat();
00480     unsigned int getUint();
00481     int getInt();
00483
00484 #if 0
00485     // Not implemented yet...
00486
00487     // The getFloat functions can be overloaded by parameter type.
00488     void getFloat(float* f);
00489     void getFloat(float* f1, float* f2);
00490     void getFloat(optix::float2* f);
00491     void getFloat(float* f1, float* f2, float* f3);
00492     void getFloat(optix::float3* f);
00493     void getFloat(float* f1, float* f2, float* f3, float* f4);
00494     void getFloat(optix::float4* f);
00495     // This one will need a different name to distinquish it from 'float getFloat
      ()'.
00496     optix::float2 getFloat2();
00497     optix::float3 getFloat3();
00498     optix::float4 getFloat4();
00499
00500     void get1fv(float* f);
00501     void get2fv(float* f);
00502     void get3fv(float* f);
00503     void get4fv(float* f);
00504
00505     get1i (int* i1);
00506     get2i (int* i1, int* i2);
00507     get3i (int* i1, int* i2, int* i3);
00508     get4i (int* i1, int* i2, int* i3, int* i4);
```

```
00509     get1iv(int* i);
00510     get2iv(int* i);
00511     get3iv(int* i);
00512     get4iv(int* i);
00513 
00514     get1ui (unsigned int* u1);
00515     get2ui (unsigned int* u1, unsigned int* u2);
00516     get3ui (unsigned int* u1, unsigned int* u2, unsigned int* u3);
00517     get4ui (unsigned int* u1, unsigned int* u2, unsigned int* u3, unsigned int* u
      4);
00518     get1uiv(unsigned int* u);
00519     get2uiv(unsigned int* u);
00520     get3uiv(unsigned int* u);
00521     get4uiv(unsigned int* u);
00522 
00523     getMatrix2x2fv(bool transpose, float* m);
00524     getMatrix2x3fv(bool transpose, float* m);
00525     getMatrix2x4fv(bool transpose, float* m);
00526     getMatrix3x2fv(bool transpose, float* m);
00527     getMatrix3x3fv(bool transpose, float* m);
00528     getMatrix3x4fv(bool transpose, float* m);
00529     getMatrix4x2fv(bool transpose, float* m);
00530     getMatrix4x3fv(bool transpose, float* m);
00531     getMatrix4x4fv(bool transpose, float* m);
00532 #endif
00533 
00534 
00537 
00538     void setBuffer(Buffer buffer);
00539     void set(Buffer buffer);
00540     void setTextureSampler(TextureSampler texturesample);
00541     void set(TextureSampler texturesample);
00542     void set(GeometryGroup group);
00543     void set(Group group);
00544     void set(Selector selector);
00545     void set(Transform transform);
00547 
00550 
00551     Buffer getBuffer();
00552     TextureSampler getTextureSampler();
00554 
00556 
00557 
00558     void setUserData(RTsize size, const void* ptr);
00560     void getUserData(RTsize size,       void* ptr);
00562 
00564     std::string getName();
00565 
00567     std::string getAnnotation();
00568 
00570     RTobjecttype getType();
00571 
00573     RTvariable get();
00574 
00576     RTsize getSize();
00577 
00578   private:
00579     typedef RTvariable api_t;
00580 
00581     RTvariable m_variable;
00582     VariableObj(RTvariable variable) : m_variable(variable) {}
00583     friend class Handle<VariableObj>;
00584 
00585   };
00586 
00587   template<class T>
00588   Handle<VariableObj> Handle<T>::operator[](const std::string& varname)
```

```
00589   {
00590     Variable v = ptr->queryVariable( varname );
00591     if( v.operator->() == 0)
00592       v = ptr->declareVariable( varname );
00593     return v;
00594   }
00595
00596   template<class T>
00597   Handle<VariableObj> Handle<T>::operator[](const char* varname)
00598   {
00599     return (*this)[ std::string( varname ) ];
00600   }
00601
00602
00603   //------------------------------------------------------------------------
00604
00605
00609   class ContextObj : public ScopedObj {
00610   public:
00611
00613     static unsigned int getDeviceCount();
00614
00616     static Context create();
00617
00619     void destroy();
00620
00622     void validate();
00623
00626     Context getContext();
00627
00630     void checkError(RTresult code);
00631
00633     std::string getErrorString( RTresult code );
00635
00638     Acceleration createAcceleration(const char* builder, const char* traverser);
00639
00641     Buffer createBuffer(unsigned int type);
00643     Buffer createBuffer(unsigned int type, RTformat format);
00646     Buffer createBuffer(unsigned int type, RTformat format, RTsize width);
00649     Buffer createBuffer(unsigned int type, RTformat format, RTsize width, RTsize
      height);
00652     Buffer createBuffer(unsigned int type, RTformat format, RTsize width, RTsize
      height, RTsize depth);
00653
00655     Buffer createBufferFromGLBO(unsigned int type, unsigned int vbo);
00656
00658     TextureSampler createTextureSamplerFromGLImage(unsigned int id, RTgltarget ta
      rget );
00659
00660 #ifdef _WIN32
00661
00662     Buffer createBufferFromD3D9Resource(unsigned int type, IDirect3DResource9 *pR
      esource);
00664     Buffer createBufferFromD3D10Resource(unsigned int type, ID3D10Resource *pReso
      urce);
00666     Buffer createBufferFromD3D11Resource(unsigned int type, ID3D11Resource *pReso
      urce);
00667
00669     TextureSampler createTextureSamplerFromD3D9Resource(IDirect3DResource9 *pReso
      urce);
00671     TextureSampler createTextureSamplerFromD3D10Resource(ID3D10Resource *pResourc
      e);
00673     TextureSampler createTextureSamplerFromD3D11Resource(ID3D11Resource *pResourc
      e);
00674 #endif
00675
00677     Geometry createGeometry();
```

```
00679     GeometryInstance createGeometryInstance();
00682     template<class Iterator>
00683     GeometryInstance createGeometryInstance( Geometry geometry, Iterator matlbegi
      n, Iterator matlend);
00684
00686     Group createGroup();
00689     template<class Iterator>
00690     Group createGroup( Iterator childbegin, Iterator childend );
00691
00693     GeometryGroup createGeometryGroup();
00696     template<class Iterator>
00697     GeometryGroup createGeometryGroup( Iterator childbegin, Iterator childend );
00698
00700     Transform createTransform();
00701
00703     Material createMaterial();
00704
00706     Program createProgramFromPTXFile   ( const std::string& ptx, const std::string
      & program_name );
00708     Program createProgramFromPTXString( const std::string& ptx, const std::string
      & program_name );
00709
00711     Selector createSelector();
00712
00714     TextureSampler createTextureSampler();
00716
00719     template<class Iterator>
00720     void setDevices(Iterator begin, Iterator end);
00721
00722 #ifdef _WIN32
00723
00724     void setD3D9Device(IDirect3DDevice9* device);
00726     void setD3D10Device(ID3D10Device* device);
00728     void setD3D11Device(ID3D11Device* device);
00729 #endif
00730
00732     std::vector<int> getEnabledDevices();
00733
00736     unsigned int getEnabledDeviceCount();
00738
00741     int getMaxTextureCount();
00742
00744     RTsize getAvailableDeviceMemory(int ordinal);
00746
00749     void setStackSize(RTsize  stack_size_bytes);
00751     RTsize getStackSize();
00752
00754     void setEntryPointCount(unsigned int  num_entry_points);
00756     unsigned int getEntryPointCount();
00757
00759     void setRayTypeCount(unsigned int  num_ray_types);
00761     unsigned int getRayTypeCount();
00763
00766     void setRayGenerationProgram(unsigned int entry_point_index, Program  program
      );
00768     Program getRayGenerationProgram(unsigned int entry_point_index);
00769
00771     void setExceptionProgram(unsigned int entry_point_index, Program  program);
00773     Program getExceptionProgram(unsigned int entry_point_index);
00774
00776     void setExceptionEnabled( RTexception exception, bool enabled );
00778     bool getExceptionEnabled( RTexception exception );
00779
00781     void setMissProgram(unsigned int ray_type_index, Program  program);
00783     Program getMissProgram(unsigned int ray_type_index);
00785
00787     void compile();
```

```
00788
00791     void launch(unsigned int entry_point_index, RTsize image_width);
00793     void launch(unsigned int entry_point_index, RTsize image_width, RTsize image_
      height);
00795     void launch(unsigned int entry_point_index, RTsize image_width, RTsize image_
      height, RTsize image_depth);
00797
00799     int getRunningState();
00800
00803     void setPrintEnabled(bool enabled);
00805     bool getPrintEnabled();
00807     void setPrintBufferSize(RTsize buffer_size_bytes);
00809     RTsize getPrintBufferSize();
00811     void setPrintLaunchIndex(int x, int y=-1, int z=-1);
00813     optix::int3 getPrintLaunchIndex();
00815
00817     Variable declareVariable (const std::string& name);
00818     Variable queryVariable   (const std::string& name);
00819     void     removeVariable  (Variable v);
00820     unsigned int getVariableCount();
00821     Variable getVariable     (unsigned int index);
00823
00825     RTcontext get();
00826   private:
00827     typedef RTcontext api_t;
00828
00829     virtual ~ContextObj() {}
00830     RTcontext m_context;
00831     ContextObj(RTcontext context) : m_context(context) {}
00832     friend class Handle<ContextObj>;
00833   };
00834
00835
00836   //------------------------------------------------------------------------------
00837
00838
00842   class ProgramObj : public ScopedObj {
00843   public:
00844     void destroy();
00845     void validate();
00846
00847     Context getContext();
00848
00849     Variable declareVariable (const std::string& name);
00850     Variable queryVariable   (const std::string& name);
00851     void     removeVariable  (Variable v);
00852     unsigned int getVariableCount();
00853     Variable getVariable     (unsigned int index);
00854
00855     RTprogram get();
00856   private:
00857     typedef RTprogram api_t;
00858     virtual ~ProgramObj() {}
00859     RTprogram m_program;
00860     ProgramObj(RTprogram program) : m_program(program) {}
00861     friend class Handle<ProgramObj>;
00862   };
00863
00864
00865   //------------------------------------------------------------------------------
00866
00867
00871   class GroupObj : public DestroyableObj {
00872   public:
00873     void destroy();
00874     void validate();
00875
```

```
00876     Context getContext();
00877
00880     void setAcceleration(Acceleration acceleration);
00882     Acceleration getAcceleration();
00884
00887     void setChildCount(unsigned int  count);
00889     unsigned int getChildCount();
00890
00892     template< typename T > void setChild(unsigned int index, T child);
00894     template< typename T > T getChild(unsigned int index);
00896
00898     RTgroup get();
00899
00900   private:
00901     typedef RTgroup api_t;
00902     virtual ~GroupObj() {}
00903     RTgroup m_group;
00904     GroupObj(RTgroup group) : m_group(group) {}
00905     friend class Handle<GroupObj>;
00906   };
00907
00908
00909   //------------------------------------------------------------------------
00910
00911
00915   class GeometryGroupObj : public DestroyableObj {
00916   public:
00917     void destroy();
00918     void validate();
00919     Context getContext();
00920
00923     void setAcceleration(Acceleration acceleration);
00925     Acceleration getAcceleration();
00927
00930     void setChildCount(unsigned int  count);
00932     unsigned int getChildCount();
00933
00935     void setChild(unsigned int index, GeometryInstance geometryinstance);
00937     GeometryInstance getChild(unsigned int index);
00939
00941     RTgeometrygroup get();
00942
00943   private:
00944     typedef RTgeometrygroup api_t;
00945     virtual ~GeometryGroupObj() {}
00946     RTgeometrygroup m_geometrygroup;
00947     GeometryGroupObj(RTgeometrygroup geometrygroup) : m_geometrygroup(geometrygro
      up) {}
00948     friend class Handle<GeometryGroupObj>;
00949   };
00950
00951
00952   //------------------------------------------------------------------------
00953
00954
00958   class TransformObj  : public DestroyableObj {
00959   public:
00960     void destroy();
00961     void validate();
00962     Context getContext();
00963
00966     template< typename T > void setChild(T child);
00968     template< typename T > T getChild();
00970
00973     void setMatrix(bool transpose, const float* matrix, const float* inverse_matr
      ix);
00975     void getMatrix(bool transpose, float* matrix, float* inverse_matrix);
```

```
00977
00979     RTtransform get();
00980
00981   private:
00982     typedef RTtransform api_t;
00983     virtual ~TransformObj() {}
00984     RTtransform m_transform;
00985     TransformObj(RTtransform transform) : m_transform(transform) {}
00986     friend class Handle<TransformObj>;
00987   };
00988
00989
00990   //------------------------------------------------------------------------
00991
00992
00996   class SelectorObj : public DestroyableObj {
00997   public:
00998     void destroy();
00999     void validate();
01000     Context getContext();
01001
01004     void setVisitProgram(Program  program);
01006     Program getVisitProgram();
01008
01011     void setChildCount(unsigned int  count);
01013     unsigned int getChildCount();
01014
01016     template< typename T > void setChild(unsigned int index, T child);
01018     template< typename T > T getChild(unsigned int index);
01020
01022     Variable declareVariable (const std::string& name);
01023     Variable queryVariable   (const std::string& name);
01024     void     removeVariable  (Variable v);
01025     unsigned int getVariableCount();
01026     Variable getVariable     (unsigned int index);
01028
01030     RTselector get();
01031
01032   private:
01033     typedef RTselector api_t;
01034     virtual ~SelectorObj() {}
01035     RTselector m_selector;
01036     SelectorObj(RTselector selector) : m_selector(selector) {}
01037     friend class Handle<SelectorObj>;
01038   };
01039
01040
01041   //------------------------------------------------------------------------
01042
01043
01047   class AccelerationObj : public DestroyableObj {
01048   public:
01049     void destroy();
01050     void validate();
01051     Context getContext();
01052
01055     void markDirty();
01057     bool isDirty();
01059
01063     void        setProperty( const std::string& name, const std::string& value );

01066     std::string getProperty( const std::string& name );
01067
01069     void        setBuilder(const std::string& builder);
01071     std::string getBuilder();
01073     void        setTraverser(const std::string& traverser);
01075     std::string getTraverser();
```

```
01077
01080     RTsize getDataSize();
01082     void   getData( void* data );
01084     void   setData( const void* data, RTsize size );
01086
01088     RTacceleration get();
01089
01090   private:
01091     typedef RTacceleration api_t;
01092     virtual ~AccelerationObj() {}
01093     RTacceleration m_acceleration;
01094     AccelerationObj(RTacceleration acceleration) : m_acceleration(acceleration) {
      }
01095     friend class Handle<AccelerationObj>;
01096   };
01097
01098
01099   //------------------------------------------------------------------------------
01100
01101
01106   class GeometryInstanceObj : public ScopedObj {
01107   public:
01108     void destroy();
01109     void validate();
01110     Context getContext();
01111
01114     void setGeometry(Geometry  geometry);
01116     Geometry getGeometry();
01117
01119     void setMaterialCount(unsigned int  count);
01121     unsigned int getMaterialCount();
01122
01124     void setMaterial(unsigned int idx, Material  material);
01126     Material getMaterial(unsigned int idx);
01127
01129     unsigned int addMaterial(Material material);
01131
01133     Variable declareVariable (const std::string& name);
01134     Variable queryVariable   (const std::string& name);
01135     void     removeVariable  (Variable v);
01136     unsigned int getVariableCount();
01137     Variable getVariable     (unsigned int index);
01139
01141     RTgeometryinstance get();
01142
01143   private:
01144     typedef RTgeometryinstance api_t;
01145     virtual ~GeometryInstanceObj() {}
01146     RTgeometryinstance m_geometryinstance;
01147     GeometryInstanceObj(RTgeometryinstance geometryinstance) : m_geometryinstance
    (geometryinstance) {}
01148     friend class Handle<GeometryInstanceObj>;
01149   };
01150
01151
01152   //------------------------------------------------------------------------------
01153
01154
01158   class GeometryObj : public ScopedObj {
01159   public:
01160     void destroy();
01161     void validate();
01162     Context getContext();
01163
01166     void markDirty();
01168     bool isDirty();
01170
```

```
01174     void setPrimitiveCount(unsigned int  num_primitives);
01177     unsigned int getPrimitiveCount();
01179
01182     void setBoundingBoxProgram(Program  program);
01184     Program getBoundingBoxProgram();
01185
01187     void setIntersectionProgram(Program  program);
01189     Program getIntersectionProgram();
01191
01193     Variable declareVariable (const std::string& name);
01194     Variable queryVariable   (const std::string& name);
01195     void     removeVariable  (Variable v);
01196     unsigned int getVariableCount();
01197     Variable getVariable     (unsigned int index);
01199
01201     RTgeometry get();
01202
01203   private:
01204     typedef RTgeometry api_t;
01205     virtual ~GeometryObj() {}
01206     RTgeometry m_geometry;
01207     GeometryObj(RTgeometry geometry) : m_geometry(geometry) {}
01208     friend class Handle<GeometryObj>;
01209   };
01210
01211
01212   //------------------------------------------------------------------------
01213
01214
01218   class MaterialObj : public ScopedObj {
01219   public:
01220     void destroy();
01221     void validate();
01222     Context getContext();
01223
01226     void setClosestHitProgram(unsigned int ray_type_index, Program  program);
01228     Program getClosestHitProgram(unsigned int ray_type_index);
01229
01231     void setAnyHitProgram(unsigned int ray_type_index, Program  program);
01233     Program getAnyHitProgram(unsigned int ray_type_index);
01235
01237     Variable declareVariable (const std::string& name);
01238     Variable queryVariable   (const std::string& name);
01239     void     removeVariable  (Variable v);
01240     unsigned int getVariableCount();
01241     Variable getVariable     (unsigned int index);
01243
01245     RTmaterial get();
01246   private:
01247     typedef RTmaterial api_t;
01248     virtual ~MaterialObj() {}
01249     RTmaterial m_material;
01250     MaterialObj(RTmaterial material) : m_material(material) {}
01251     friend class Handle<MaterialObj>;
01252   };
01253
01254
01255   //------------------------------------------------------------------------
01256
01257
01261   class TextureSamplerObj : public DestroyableObj {
01262   public:
01263     void destroy();
01264     void validate();
01265     Context getContext();
01266
01269     void setMipLevelCount (unsigned int  num_mip_levels);
```

```
01271     unsigned int getMipLevelCount ();
01272
01274     void setArraySize(unsigned int  num_textures_in_array);
01276     unsigned int getArraySize();
01277
01279     void setWrapMode(unsigned int dim, RTwrapmode wrapmode);
01281     RTwrapmode getWrapMode(unsigned int dim);
01282
01284     void setFilteringModes(RTfiltermode  minification, RTfiltermode  magnificatio
      n, RTfiltermode  mipmapping);
01286     void getFilteringModes(RTfiltermode& minification, RTfiltermode& magnificatio
      n, RTfiltermode& mipmapping);
01287
01289     void setMaxAnisotropy(float value);
01291     float getMaxAnisotropy();
01292
01294     void setReadMode(RTtexturereadmode  readmode);
01296     RTtexturereadmode getReadMode();
01297
01299     void setIndexingMode(RTtextureindexmode  indexmode);
01301     RTtextureindexmode getIndexingMode();
01303
01306     void setBuffer(unsigned int texture_array_idx, unsigned int mip_level,
      Buffer buffer);
01308     Buffer getBuffer(unsigned int texture_array_idx, unsigned int mip_level);
01310
01312     RTtexturesampler get();
01313
01316     void registerGLTexture();
01318     void unregisterGLTexture();
01320
01321 #ifdef _WIN32
01322
01325     void registerD3D9Texture();
01327     void registerD3D10Texture();
01329     void registerD3D11Texture();
01330
01332     void unregisterD3D9Texture();
01334     void unregisterD3D10Texture();
01336     void unregisterD3D11Texture();
01338
01339 #endif
01340
01341   private:
01342     typedef RTtexturesampler api_t;
01343     virtual ~TextureSamplerObj() {}
01344     RTtexturesampler m_texturesampler;
01345     TextureSamplerObj(RTtexturesampler texturesampler) : m_texturesampler(texture
      sampler) {}
01346     friend class Handle<TextureSamplerObj>;
01347   };
01348
01349
01350   //------------------------------------------------------------------------------
01351
01352
01356   class BufferObj : public DestroyableObj {
01357   public:
01358     void destroy();
01359     void validate();
01360     Context getContext();
01361
01364     void setFormat    (RTformat  format);
01366     RTformat getFormat();
01367
01369     void setElementSize  (RTsize size_of_element);
01371     RTsize getElementSize();
```

```
01372
01374     void setSize(RTsize  width);
01376     void getSize(RTsize& width);
01378     void setSize(RTsize  width, RTsize  height);
01380     void getSize(RTsize& width, RTsize& height);
01383     void setSize(RTsize  width, RTsize  height, RTsize  depth);
01385     void getSize(RTsize& width, RTsize& height, RTsize& depth);
01386
01388     void setSize(unsigned int dimensionality, const RTsize* dims);
01390     void getSize(unsigned int dimensionality,       RTsize* dims);
01391
01393     unsigned int getDimensionality();
01395
01398     unsigned int getGLBOId();
01399
01401     void registerGLBuffer();
01403     void unregisterGLBuffer();
01405
01406 #ifdef _WIN32
01407
01410     void registerD3D9Buffer();
01412     void registerD3D10Buffer();
01414     void registerD3D11Buffer();
01415
01417     void unregisterD3D9Buffer();
01419     void unregisterD3D10Buffer();
01421     void unregisterD3D11Buffer();
01422
01424     IDirect3DResource9* getD3D9Resource();
01426     ID3D10Resource* getD3D10Resource();
01428     ID3D11Resource* getD3D11Resource();
01430
01431 #endif
01432
01435     void* map();
01437     void unmap();
01439
01441     RTbuffer get();
01442
01443   private:
01444     typedef RTbuffer api_t;
01445     virtual ~BufferObj() {}
01446     RTbuffer m_buffer;
01447     BufferObj(RTbuffer buffer) : m_buffer(buffer) {}
01448     friend class Handle<BufferObj>;
01449   };
01450
01451
01452   //------------------------------------------------------------------------
01453
01454
01455   inline void APIObj::checkError( RTresult code )
01456   {
01457     if( code != RT_SUCCESS) {
01458       RTcontext c = this->getContext()->get();
01459       throw Exception::makeException( code, c );
01460     }
01461   }
01462
01463   inline void APIObj::checkErrorNoGetContext( RTresult code )
01464   {
01465     if( code != RT_SUCCESS) {
01466       throw Exception::makeException( code, 0u );
01467     }
01468   }
01469
01470   inline Context ContextObj::getContext()
```

```
01471   {
01472     return Context::take( m_context );
01473   }
01474
01475   inline void ContextObj::checkError(RTresult code)
01476   {
01477     if( code != RT_SUCCESS)
01478       throw Exception::makeException( code, m_context );
01479   }
01480
01481   inline unsigned int ContextObj::getDeviceCount()
01482   {
01483     unsigned int count;
01484     if( RTresult code = rtDeviceGetDeviceCount(&count) )
01485       throw Exception::makeException( code, 0 );
01486
01487     return count;
01488   }
01489
01490
01491   inline Context ContextObj::create()
01492   {
01493     RTcontext c;
01494     if( RTresult code = rtContextCreate(&c) )
01495       throw Exception::makeException( code, 0 );
01496
01497     return Context::take(c);
01498   }
01499
01500   inline void ContextObj::destroy()
01501   {
01502     checkError( rtContextDestroy( m_context ) );
01503     m_context = 0;
01504   }
01505
01506   inline void ContextObj::validate()
01507   {
01508     checkError( rtContextValidate( m_context ) );
01509   }
01510
01511   inline Acceleration ContextObj::createAcceleration(const char* builder, const c
      har* traverser)
01512   {
01513     RTacceleration acceleration;
01514     checkError( rtAccelerationCreate( m_context, &acceleration ) );
01515     checkError( rtAccelerationSetBuilder( acceleration, builder ) );
01516     checkError( rtAccelerationSetTraverser( acceleration, traverser ) );
01517     return Acceleration::take(acceleration);
01518   }
01519
01520
01521   inline Buffer ContextObj::createBuffer(unsigned int type)
01522   {
01523     RTbuffer buffer;
01524     checkError( rtBufferCreate( m_context, type, &buffer ) );
01525     return Buffer::take(buffer);
01526   }
01527
01528   inline Buffer ContextObj::createBuffer(unsigned int type, RTformat format)
01529   {
01530     RTbuffer buffer;
01531     checkError( rtBufferCreate( m_context, type, &buffer ) );
01532     checkError( rtBufferSetFormat( buffer, format ) );
01533     return Buffer::take(buffer);
01534   }
01535
01536   inline Buffer ContextObj::createBuffer(unsigned int type, RTformat format, RTsi
```

```
      ze width)
01537   {
01538     RTbuffer buffer;
01539     checkError( rtBufferCreate( m_context, type, &buffer ) );
01540     checkError( rtBufferSetFormat( buffer, format ) );
01541     checkError( rtBufferSetSize1D( buffer, width ) );
01542     return Buffer::take(buffer);
01543   }
01544
01545   inline Buffer ContextObj::createBuffer(unsigned int type, RTformat format, RTsi
      ze width, RTsize height)
01546   {
01547     RTbuffer buffer;
01548     checkError( rtBufferCreate( m_context, type, &buffer ) );
01549     checkError( rtBufferSetFormat( buffer, format ) );
01550     checkError( rtBufferSetSize2D( buffer, width, height ) );
01551     return Buffer::take(buffer);
01552   }
01553
01554   inline Buffer ContextObj::createBuffer(unsigned int type, RTformat format, RTsi
      ze width, RTsize height, RTsize depth)
01555   {
01556     RTbuffer buffer;
01557     checkError( rtBufferCreate( m_context, type, &buffer ) );
01558     checkError( rtBufferSetFormat( buffer, format ) );
01559     checkError( rtBufferSetSize3D( buffer, width, height, depth ) );
01560     return Buffer::take(buffer);
01561   }
01562
01563   inline Buffer ContextObj::createBufferFromGLBO(unsigned int type, unsigned int
      vbo)
01564   {
01565     RTbuffer buffer;
01566     checkError( rtBufferCreateFromGLBO( m_context, type, vbo, &buffer ) );
01567     return Buffer::take(buffer);
01568   }
01569
01570 #ifdef _WIN32
01571
01572   inline Buffer ContextObj::createBufferFromD3D9Resource(unsigned int type, IDire
      ct3DResource9 *pResource)
01573   {
01574     RTbuffer buffer;
01575     checkError( rtBufferCreateFromD3D9Resource( m_context, type, pResource, &buff
      er ) );
01576     return Buffer::take(buffer);
01577   }
01578
01579   inline Buffer ContextObj::createBufferFromD3D10Resource(unsigned int type, ID3D
      10Resource *pResource)
01580   {
01581     RTbuffer buffer;
01582     checkError( rtBufferCreateFromD3D10Resource( m_context, type, pResource, &buf
      fer ) );
01583     return Buffer::take(buffer);
01584   }
01585
01586   inline Buffer ContextObj::createBufferFromD3D11Resource(unsigned int type, ID3D
      11Resource *pResource)
01587   {
01588     RTbuffer buffer;
01589     checkError( rtBufferCreateFromD3D11Resource( m_context, type, pResource, &buf
      fer ) );
01590     return Buffer::take(buffer);
01591   }
01592
01593   inline TextureSampler ContextObj::createTextureSamplerFromD3D9Resource(IDirect3
```

```
      DResource9 *pResource)
01594   {
01595     RTtexturesampler textureSampler;
01596     checkError( rtTextureSamplerCreateFromD3D9Resource(m_context, pResource, &tex
      tureSampler));
01597     return TextureSampler::take(textureSampler);
01598   }
01599
01600   inline TextureSampler ContextObj::createTextureSamplerFromD3D10Resource(ID3D10R
      esource *pResource)
01601   {
01602     RTtexturesampler textureSampler;
01603     checkError( rtTextureSamplerCreateFromD3D10Resource(m_context, pResource, &te
      xtureSampler));
01604     return TextureSampler::take(textureSampler);
01605   }
01606
01607   inline TextureSampler ContextObj::createTextureSamplerFromD3D11Resource(ID3D11R
      esource *pResource)
01608   {
01609     RTtexturesampler textureSampler;
01610     checkError( rtTextureSamplerCreateFromD3D11Resource(m_context, pResource, &te
      xtureSampler));
01611     return TextureSampler::take(textureSampler);
01612   }
01613
01614   inline void ContextObj::setD3D9Device(IDirect3DDevice9* device)
01615   {
01616     checkError( rtContextSetD3D9Device( m_context, device ) );
01617   }
01618
01619   inline void ContextObj::setD3D10Device(ID3D10Device* device)
01620   {
01621     checkError( rtContextSetD3D10Device( m_context, device ) );
01622   }
01623
01624   inline void ContextObj::setD3D11Device(ID3D11Device* device)
01625   {
01626     checkError( rtContextSetD3D11Device( m_context, device ) );
01627   }
01628
01629 #endif
01630
01631   inline TextureSampler ContextObj::createTextureSamplerFromGLImage(unsigned int
      id, RTgltarget target)
01632   {
01633     RTtexturesampler textureSampler;
01634     checkError( rtTextureSamplerCreateFromGLImage(m_context, id, target, &texture
      Sampler));
01635     return TextureSampler::take(textureSampler);
01636   }
01637
01638   inline Geometry ContextObj::createGeometry()
01639   {
01640     RTgeometry geometry;
01641     checkError( rtGeometryCreate( m_context, &geometry ) );
01642     return Geometry::take(geometry);
01643   }
01644
01645   inline GeometryInstance ContextObj::createGeometryInstance()
01646   {
01647     RTgeometryinstance geometryinstance;
01648     checkError( rtGeometryInstanceCreate( m_context, &geometryinstance ) );
01649     return GeometryInstance::take(geometryinstance);
01650   }
01651
01652   template<class Iterator>
```

```
01653     GeometryInstance ContextObj::createGeometryInstance( Geometry geometry, Itera
      tor matlbegin, Iterator matlend)
01654   {
01655     GeometryInstance result = createGeometryInstance();
01656     result->setGeometry( geometry );
01657     unsigned int count = 0;
01658     for( Iterator iter = matlbegin; iter != matlend; ++iter )
01659       ++count;
01660     result->setMaterialCount( count );
01661     unsigned int index = 0;
01662     for(Iterator iter = matlbegin; iter != matlend; ++iter, ++index )
01663       result->setMaterial( index, *iter );
01664     return result;
01665   }
01666
01667   inline Group ContextObj::createGroup()
01668   {
01669     RTgroup group;
01670     checkError( rtGroupCreate( m_context, &group ) );
01671     return Group::take(group);
01672   }
01673
01674   template<class Iterator>
01675     inline Group ContextObj::createGroup( Iterator childbegin, Iterator childend
      )
01676   {
01677     Group result = createGroup();
01678     unsigned int count = 0;
01679     for(Iterator iter = childbegin; iter != childend; ++iter )
01680       ++count;
01681     result->setChildCount( count );
01682     unsigned int index = 0;
01683     for(Iterator iter = childbegin; iter != childend; ++iter, ++index )
01684       result->setChild( index, *iter );
01685     return result;
01686   }
01687
01688   inline GeometryGroup ContextObj::createGeometryGroup()
01689   {
01690     RTgeometrygroup gg;
01691     checkError( rtGeometryGroupCreate( m_context, &gg ) );
01692     return GeometryGroup::take( gg );
01693   }
01694
01695   template<class Iterator>
01696   inline GeometryGroup ContextObj::createGeometryGroup( Iterator childbegin, Iter
      ator childend )
01697   {
01698     GeometryGroup result = createGeometryGroup();
01699     unsigned int count = 0;
01700     for(Iterator iter = childbegin; iter != childend; ++iter )
01701       ++count;
01702     result->setChildCount( count );
01703     unsigned int index = 0;
01704     for(Iterator iter = childbegin; iter != childend; ++iter, ++index )
01705       result->setChild( index, *iter );
01706     return result;
01707   }
01708
01709   inline Transform ContextObj::createTransform()
01710   {
01711     RTtransform t;
01712     checkError( rtTransformCreate( m_context, &t ) );
01713     return Transform::take( t );
01714   }
01715
01716   inline Material ContextObj::createMaterial()
```

```
01717   {
01718     RTmaterial material;
01719     checkError( rtMaterialCreate( m_context, &material ) );
01720     return Material::take(material);
01721   }
01722
01723   inline Program ContextObj::createProgramFromPTXFile( const std::string& filenam
      e, const std::string& program_name )
01724   {
01725     RTprogram program;
01726     checkError( rtProgramCreateFromPTXFile( m_context, filename.c_str(), program_
      name.c_str(), &program ) );
01727     return Program::take(program);
01728   }
01729
01730   inline Program ContextObj::createProgramFromPTXString( const std::string& ptx,
      const std::string& program_name )
01731   {
01732     RTprogram program;
01733     checkError( rtProgramCreateFromPTXString( m_context, ptx.c_str(), program_nam
      e.c_str(), &program ) );
01734     return Program::take(program);
01735   }
01736
01737   inline Selector ContextObj::createSelector()
01738   {
01739     RTselector selector;
01740     checkError( rtSelectorCreate( m_context, &selector ) );
01741     return Selector::take(selector);
01742   }
01743
01744   inline TextureSampler ContextObj::createTextureSampler()
01745   {
01746     RTtexturesampler texturesampler;
01747     checkError( rtTextureSamplerCreate( m_context, &texturesampler ) );
01748     return TextureSampler::take(texturesampler);
01749   }
01750
01751   inline std::string ContextObj::getErrorString( RTresult code )
01752   {
01753     const char* str;
01754     rtContextGetErrorString( m_context, code, &str);
01755     return std::string(str);
01756   }
01757
01758   template<class Iterator> inline
01759     void ContextObj::setDevices(Iterator begin, Iterator end)
01760   {
01761     std::vector<int> devices;
01762     std::copy( begin, end, std::insert_iterator<std::vector<int> >( devices, devi
      ces.begin() ) );
01763     checkError( rtContextSetDevices( m_context, static_cast<unsigned int>(devices
      .size()), &devices[0]) );
01764   }
01765
01766   inline std::vector<int> ContextObj::getEnabledDevices()
01767   {
01768     // Initialize with the number of enabled devices
01769     std::vector<int> devices(getEnabledDeviceCount());
01770     checkError( rtContextGetDevices( m_context, &devices[0] ) );
01771     return devices;
01772   }
01773
01774   inline unsigned int ContextObj::getEnabledDeviceCount()
01775   {
01776     unsigned int num;
01777     checkError( rtContextGetDeviceCount( m_context, &num ) );
```

```
01778     return num;
01779   }
01780
01781   inline int ContextObj::getMaxTextureCount()
01782   {
01783     int tex_count;
01784     checkError( rtContextGetAttribute( m_context, RT_CONTEXT_ATTRIBUTE_MAX_TEXTUR
      E_COUNT, sizeof(tex_count), &tex_count) );
01785     return tex_count;
01786   }
01787
01788   inline RTsize ContextObj::getAvailableDeviceMemory(int ordinal)
01789   {
01790     RTsize free_mem;
01791     checkError( rtContextGetAttribute( m_context,
01792                                        static_cast<RTcontextattribute>(RT_CONTEXT
      _ATTRIBUTE_AVAILABLE_DEVICE_MEMORY + ordinal),
01793                                        sizeof(free_mem), &free_mem) );
01794     return free_mem;
01795   }
01796
01797   inline void ContextObj::setStackSize(RTsize  stack_size_bytes)
01798   {
01799     checkError(rtContextSetStackSize(m_context, stack_size_bytes) );
01800   }
01801
01802   inline RTsize ContextObj::getStackSize()
01803   {
01804     RTsize result;
01805     checkError( rtContextGetStackSize( m_context, &result ) );
01806     return result;
01807   }
01808
01809   inline void ContextObj::setEntryPointCount(unsigned int  num_entry_points)
01810   {
01811     checkError( rtContextSetEntryPointCount( m_context, num_entry_points ) );
01812   }
01813
01814   inline unsigned int ContextObj::getEntryPointCount()
01815   {
01816     unsigned int result;
01817     checkError( rtContextGetEntryPointCount( m_context, &result ) );
01818     return result;
01819   }
01820
01821
01822   inline void ContextObj::setRayGenerationProgram(unsigned int entry_point_index,
       Program  program)
01823   {
01824     checkError( rtContextSetRayGenerationProgram( m_context, entry_point_index, p
      rogram->get() ) );
01825   }
01826
01827   inline Program ContextObj::getRayGenerationProgram(unsigned int entry_point_ind
      ex)
01828   {
01829     RTprogram result;
01830     checkError( rtContextGetRayGenerationProgram( m_context, entry_point_index, &
      result ) );
01831     return Program::take( result );
01832   }
01833
01834
01835   inline void ContextObj::setExceptionProgram(unsigned int entry_point_index,
      Program  program)
01836   {
01837     checkError( rtContextSetExceptionProgram( m_context, entry_point_index, progr
```

```
      am->get() ) );
01838   }
01839
01840   inline Program ContextObj::getExceptionProgram(unsigned int entry_point_index)
01841   {
01842     RTprogram result;
01843     checkError( rtContextGetExceptionProgram( m_context, entry_point_index, &resu
      lt ) );
01844     return Program::take( result );
01845   }
01846
01847
01848   inline void ContextObj::setExceptionEnabled( RTexception exception, bool enable
      d )
01849   {
01850     checkError( rtContextSetExceptionEnabled( m_context, exception, enabled ) );
01851   }
01852
01853   inline bool ContextObj::getExceptionEnabled( RTexception exception )
01854   {
01855     int enabled;
01856     checkError( rtContextGetExceptionEnabled( m_context, exception, &enabled ) );

01857     return enabled != 0;
01858   }
01859
01860
01861   inline void ContextObj::setRayTypeCount(unsigned int  num_ray_types)
01862   {
01863     checkError( rtContextSetRayTypeCount( m_context, num_ray_types ) );
01864   }
01865
01866   inline unsigned int ContextObj::getRayTypeCount()
01867   {
01868     unsigned int result;
01869     checkError( rtContextGetRayTypeCount( m_context, &result ) );
01870     return result;
01871   }
01872
01873   inline void ContextObj::setMissProgram(unsigned int ray_type_index, Program  pr
      ogram)
01874   {
01875     checkError( rtContextSetMissProgram( m_context, ray_type_index, program->get(
      ) ) );
01876   }
01877
01878   inline Program ContextObj::getMissProgram(unsigned int ray_type_index)
01879   {
01880     RTprogram result;
01881     checkError( rtContextGetMissProgram( m_context, ray_type_index, &result ) );
01882     return Program::take( result );
01883   }
01884
01885   inline void ContextObj::compile()
01886   {
01887     checkError( rtContextCompile( m_context ) );
01888   }
01889
01890   inline void ContextObj::launch(unsigned int entry_point_index, RTsize image_wid
      th)
01891   {
01892     checkError( rtContextLaunch1D( m_context, entry_point_index, image_width ) );

01893   }
01894
01895   inline void ContextObj::launch(unsigned int entry_point_index, RTsize image_wid
      th, RTsize image_height)
```

```
01896   {
01897     checkError( rtContextLaunch2D( m_context, entry_point_index, image_width, ima
      ge_height ) );
01898   }
01899
01900   inline void ContextObj::launch(unsigned int entry_point_index, RTsize image_wid
      th, RTsize image_height, RTsize image_depth)
01901   {
01902     checkError( rtContextLaunch3D( m_context, entry_point_index, image_width, ima
      ge_height, image_depth ) );
01903   }
01904
01905
01906   inline int ContextObj::getRunningState()
01907   {
01908     int result;
01909     checkError( rtContextGetRunningState( m_context, &result ) );
01910     return result;
01911   }
01912
01913   inline void ContextObj::setPrintEnabled(bool enabled)
01914   {
01915     checkError( rtContextSetPrintEnabled( m_context, enabled ) );
01916   }
01917
01918   inline bool ContextObj::getPrintEnabled()
01919   {
01920     int enabled;
01921     checkError( rtContextGetPrintEnabled( m_context, &enabled ) );
01922     return enabled != 0;
01923   }
01924
01925   inline void ContextObj::setPrintBufferSize(RTsize buffer_size_bytes)
01926   {
01927     checkError( rtContextSetPrintBufferSize( m_context, buffer_size_bytes ) );
01928   }
01929
01930   inline RTsize ContextObj::getPrintBufferSize()
01931   {
01932     RTsize result;
01933     checkError( rtContextGetPrintBufferSize( m_context, &result ) );
01934     return result;
01935   }
01936
01937   inline void ContextObj::setPrintLaunchIndex(int x, int y, int z)
01938   {
01939     checkError( rtContextSetPrintLaunchIndex( m_context, x, y, z ) );
01940   }
01941
01942   inline optix::int3 ContextObj::getPrintLaunchIndex()
01943   {
01944     optix::int3 result;
01945     checkError( rtContextGetPrintLaunchIndex( m_context, &result.x, &result.y, &r
      esult.z ) );
01946     return result;
01947   }
01948
01949   inline Variable ContextObj::declareVariable(const std::string& name)
01950   {
01951     RTvariable v;
01952     checkError( rtContextDeclareVariable( m_context, name.c_str(), &v ) );
01953     return Variable::take( v );
01954   }
01955
01956   inline Variable ContextObj::queryVariable(const std::string& name)
01957   {
01958     RTvariable v;
```

```
01959     checkError( rtContextQueryVariable( m_context, name.c_str(), &v ) );
01960     return Variable::take( v );
01961   }
01962
01963   inline void ContextObj::removeVariable(Variable v)
01964   {
01965     checkError( rtContextRemoveVariable( m_context, v->get() ) );
01966   }
01967
01968   inline unsigned int ContextObj::getVariableCount()
01969   {
01970     unsigned int result;
01971     checkError( rtContextGetVariableCount( m_context, &result ) );
01972     return result;
01973   }
01974
01975   inline Variable ContextObj::getVariable(unsigned int index)
01976   {
01977     RTvariable v;
01978     checkError( rtContextGetVariable( m_context, index, &v ) );
01979     return Variable::take( v );
01980   }
01981
01982
01983   inline RTcontext ContextObj::get()
01984   {
01985     return m_context;
01986   }
01987
01988   inline void ProgramObj::destroy()
01989   {
01990     checkError( rtProgramDestroy( m_program ) );
01991   }
01992
01993   inline void ProgramObj::validate()
01994   {
01995     checkError( rtProgramValidate( m_program ) );
01996   }
01997
01998   inline Context ProgramObj::getContext()
01999   {
02000     RTcontext c;
02001     checkErrorNoGetContext( rtProgramGetContext( m_program, &c ) );
02002     return Context::take( c );
02003   }
02004
02005   inline Variable ProgramObj::declareVariable(const std::string& name)
02006   {
02007     RTvariable v;
02008     checkError( rtProgramDeclareVariable( m_program, name.c_str(), &v ) );
02009     return Variable::take( v );
02010   }
02011
02012   inline Variable ProgramObj::queryVariable(const std::string& name)
02013   {
02014     RTvariable v;
02015     checkError( rtProgramQueryVariable( m_program, name.c_str(), &v ) );
02016     return Variable::take( v );
02017   }
02018
02019   inline void ProgramObj::removeVariable(Variable v)
02020   {
02021     checkError( rtProgramRemoveVariable( m_program, v->get() ) );
02022   }
02023
02024   inline unsigned int ProgramObj::getVariableCount()
02025   {
```

```
02026     unsigned int result;
02027     checkError( rtProgramGetVariableCount( m_program, &result ) );
02028     return result;
02029   }
02030
02031   inline Variable ProgramObj::getVariable(unsigned int index)
02032   {
02033     RTvariable v;
02034     checkError( rtProgramGetVariable( m_program, index, &v ) );
02035     return Variable::take(v);
02036   }
02037
02038   inline RTprogram ProgramObj::get()
02039   {
02040     return m_program;
02041   }
02042
02043   inline void GroupObj::destroy()
02044   {
02045     checkError( rtGroupDestroy( m_group ) );
02046   }
02047
02048   inline void GroupObj::validate()
02049   {
02050     checkError( rtGroupValidate( m_group ) );
02051   }
02052
02053   inline Context GroupObj::getContext()
02054   {
02055     RTcontext c;
02056     checkErrorNoGetContext( rtGroupGetContext( m_group, &c) );
02057     return Context::take(c);
02058   }
02059
02060   inline void SelectorObj::destroy()
02061   {
02062     checkError( rtSelectorDestroy( m_selector ) );
02063   }
02064
02065   inline void SelectorObj::validate()
02066   {
02067     checkError( rtSelectorValidate( m_selector ) );
02068   }
02069
02070   inline Context SelectorObj::getContext()
02071   {
02072     RTcontext c;
02073     checkErrorNoGetContext( rtSelectorGetContext( m_selector, &c ) );
02074     return Context::take( c );
02075   }
02076
02077   inline void SelectorObj::setVisitProgram(Program program)
02078   {
02079     checkError( rtSelectorSetVisitProgram( m_selector, program->get() ) );
02080   }
02081
02082   inline Program SelectorObj::getVisitProgram()
02083   {
02084     RTprogram result;
02085     checkError( rtSelectorGetVisitProgram( m_selector, &result ) );
02086     return Program::take( result );
02087   }
02088
02089   inline void SelectorObj::setChildCount(unsigned int count)
02090   {
02091     checkError( rtSelectorSetChildCount( m_selector, count) );
02092   }
```

```
02093
02094   inline unsigned int SelectorObj::getChildCount()
02095   {
02096     unsigned int result;
02097     checkError( rtSelectorGetChildCount( m_selector, &result ) );
02098     return result;
02099   }
02100
02101   template< typename T >
02102   inline void SelectorObj::setChild(unsigned int index, T child)
02103   {
02104     checkError( rtSelectorSetChild( m_selector, index, child->get() ) );
02105   }
02106
02107   template< typename T >
02108   inline T SelectorObj::getChild(unsigned int index)
02109   {
02110     RTobject result;
02111     checkError( rtSelectorGetChild( m_selector, index, &result ) );
02112     return T::take( result );
02113   }
02114
02115   inline Variable SelectorObj::declareVariable(const std::string& name)
02116   {
02117     RTvariable v;
02118     checkError( rtSelectorDeclareVariable( m_selector, name.c_str(), &v ) );
02119     return Variable::take( v );
02120   }
02121
02122   inline Variable SelectorObj::queryVariable(const std::string& name)
02123   {
02124     RTvariable v;
02125     checkError( rtSelectorQueryVariable( m_selector, name.c_str(), &v ) );
02126     return Variable::take( v );
02127   }
02128
02129   inline void SelectorObj::removeVariable(Variable v)
02130   {
02131     checkError( rtSelectorRemoveVariable( m_selector, v->get() ) );
02132   }
02133
02134   inline unsigned int SelectorObj::getVariableCount()
02135   {
02136     unsigned int result;
02137     checkError( rtSelectorGetVariableCount( m_selector, &result ) );
02138     return result;
02139   }
02140
02141   inline Variable SelectorObj::getVariable(unsigned int index)
02142   {
02143     RTvariable v;
02144     checkError( rtSelectorGetVariable( m_selector, index, &v ) );
02145     return Variable::take( v );
02146   }
02147
02148   inline RTselector SelectorObj::get()
02149   {
02150     return m_selector;
02151   }
02152
02153   inline void GroupObj::setAcceleration(Acceleration acceleration)
02154   {
02155     checkError( rtGroupSetAcceleration( m_group, acceleration->get() ) );
02156   }
02157
02158   inline Acceleration GroupObj::getAcceleration()
02159   {
```

```
02160     RTacceleration result;
02161     checkError( rtGroupGetAcceleration( m_group, &result ) );
02162     return Acceleration::take( result );
02163   }
02164
02165   inline void GroupObj::setChildCount(unsigned int  count)
02166   {
02167     checkError( rtGroupSetChildCount( m_group, count ) );
02168   }
02169
02170   inline unsigned int GroupObj::getChildCount()
02171   {
02172     unsigned int result;
02173     checkError( rtGroupGetChildCount( m_group, &result ) );
02174     return result;
02175   }
02176
02177   template< typename T >
02178   inline void GroupObj::setChild(unsigned int index, T child)
02179   {
02180     checkError( rtGroupSetChild( m_group, index, child->get() ) );
02181   }
02182
02183   template< typename T >
02184   inline T GroupObj::getChild(unsigned int index)
02185   {
02186     RTobject result;
02187     checkError( rtGroupGetChild( m_group, index, &result) );
02188     return T::take( result );
02189   }
02190
02191   inline RTgroup GroupObj::get()
02192   {
02193     return m_group;
02194   }
02195
02196   inline void GeometryGroupObj::destroy()
02197   {
02198     checkError( rtGeometryGroupDestroy( m_geometrygroup ) );
02199   }
02200
02201   inline void GeometryGroupObj::validate()
02202   {
02203     checkError( rtGeometryGroupValidate( m_geometrygroup ) );
02204   }
02205
02206   inline Context GeometryGroupObj::getContext()
02207   {
02208     RTcontext c;
02209     checkErrorNoGetContext( rtGeometryGroupGetContext( m_geometrygroup, &c) );
02210     return Context::take(c);
02211   }
02212
02213   inline void GeometryGroupObj::setAcceleration(Acceleration acceleration)
02214   {
02215     checkError( rtGeometryGroupSetAcceleration( m_geometrygroup, acceleration->
      get() ) );
02216   }
02217
02218   inline Acceleration GeometryGroupObj::getAcceleration()
02219   {
02220     RTacceleration result;
02221     checkError( rtGeometryGroupGetAcceleration( m_geometrygroup, &result ) );
02222     return Acceleration::take( result );
02223   }
02224
02225   inline void GeometryGroupObj::setChildCount(unsigned int  count)
```

```
02226   {
02227     checkError( rtGeometryGroupSetChildCount( m_geometrygroup, count ) );
02228   }
02229
02230   inline unsigned int GeometryGroupObj::getChildCount()
02231   {
02232     unsigned int result;
02233     checkError( rtGeometryGroupGetChildCount( m_geometrygroup, &result ) );
02234     return result;
02235   }
02236
02237   inline void GeometryGroupObj::setChild(unsigned int index, GeometryInstance chi
      ld)
02238   {
02239     checkError( rtGeometryGroupSetChild( m_geometrygroup, index, child->get() ) )
      ;
02240   }
02241
02242   inline GeometryInstance GeometryGroupObj::getChild(unsigned int index)
02243   {
02244     RTgeometryinstance result;
02245     checkError( rtGeometryGroupGetChild( m_geometrygroup, index, &result) );
02246     return GeometryInstance::take( result );
02247   }
02248
02249   inline RTgeometrygroup GeometryGroupObj::get()
02250   {
02251     return m_geometrygroup;
02252   }
02253
02254   inline void TransformObj::destroy()
02255   {
02256     checkError( rtTransformDestroy( m_transform ) );
02257   }
02258
02259   inline void TransformObj::validate()
02260   {
02261     checkError( rtTransformValidate( m_transform ) );
02262   }
02263
02264   inline Context TransformObj::getContext()
02265   {
02266     RTcontext c;
02267     checkErrorNoGetContext( rtTransformGetContext( m_transform, &c) );
02268     return Context::take(c);
02269   }
02270
02271   template< typename T >
02272   inline void TransformObj::setChild(T child)
02273   {
02274     checkError( rtTransformSetChild( m_transform, child->get() ) );
02275   }
02276
02277   template< typename T >
02278   inline T TransformObj::getChild()
02279   {
02280     RTobject result;
02281     checkError( rtTransformGetChild( m_transform, &result) );
02282     return T::take( result );
02283   }
02284
02285   inline void TransformObj::setMatrix(bool transpose, const float* matrix, const
      float* inverse_matrix)
02286   {
02287     rtTransformSetMatrix( m_transform, transpose, matrix, inverse_matrix );
02288   }
02289
```

```
02290   inline void TransformObj::getMatrix(bool transpose, float* matrix, float* inver
      se_matrix)
02291   {
02292     rtTransformGetMatrix( m_transform, transpose, matrix, inverse_matrix );
02293   }
02294
02295   inline RTtransform TransformObj::get()
02296   {
02297     return m_transform;
02298   }
02299
02300   inline void AccelerationObj::destroy()
02301   {
02302     checkError( rtAccelerationDestroy(m_acceleration) );
02303   }
02304
02305   inline void AccelerationObj::validate()
02306   {
02307     checkError( rtAccelerationValidate(m_acceleration) );
02308   }
02309
02310   inline Context AccelerationObj::getContext()
02311   {
02312     RTcontext c;
02313     checkErrorNoGetContext( rtAccelerationGetContext(m_acceleration, &c ) );
02314     return Context::take( c );
02315   }
02316
02317   inline void AccelerationObj::markDirty()
02318   {
02319     checkError( rtAccelerationMarkDirty(m_acceleration) );
02320   }
02321
02322   inline bool AccelerationObj::isDirty()
02323   {
02324     int dirty;
02325     checkError( rtAccelerationIsDirty(m_acceleration,&dirty) );
02326     return dirty != 0;
02327   }
02328
02329   inline void AccelerationObj::setProperty( const std::string& name, const std::s
      tring& value )
02330   {
02331     checkError( rtAccelerationSetProperty(m_acceleration, name.c_str(), value.c_s
      tr() ) );
02332   }
02333
02334   inline std::string AccelerationObj::getProperty( const std::string& name )
02335   {
02336     const char* s;
02337     checkError( rtAccelerationGetProperty(m_acceleration, name.c_str(), &s ) );
02338     return std::string( s );
02339   }
02340
02341   inline void AccelerationObj::setBuilder(const std::string& builder)
02342   {
02343     checkError( rtAccelerationSetBuilder(m_acceleration, builder.c_str() ) );
02344   }
02345
02346   inline std::string AccelerationObj::getBuilder()
02347   {
02348     const char* s;
02349     checkError( rtAccelerationGetBuilder(m_acceleration, &s ) );
02350     return std::string( s );
02351   }
02352
02353   inline void AccelerationObj::setTraverser(const std::string& traverser)
```

```
02354   {
02355     checkError( rtAccelerationSetTraverser(m_acceleration, traverser.c_str() ) );
02356   }
02357
02358   inline std::string AccelerationObj::getTraverser()
02359   {
02360     const char* s;
02361     checkError( rtAccelerationGetTraverser(m_acceleration, &s ) );
02362     return std::string( s );
02363   }
02364
02365   inline RTsize AccelerationObj::getDataSize()
02366   {
02367     RTsize sz;
02368     checkError( rtAccelerationGetDataSize(m_acceleration, &sz) );
02369     return sz;
02370   }
02371
02372   inline void AccelerationObj::getData( void* data )
02373   {
02374     checkError( rtAccelerationGetData(m_acceleration,data) );
02375   }
02376
02377   inline void AccelerationObj::setData( const void* data, RTsize size )
02378   {
02379     checkError( rtAccelerationSetData(m_acceleration,data,size) );
02380   }
02381
02382   inline RTacceleration AccelerationObj::get()
02383   {
02384     return m_acceleration;
02385   }
02386
02387   inline void GeometryInstanceObj::destroy()
02388   {
02389     checkError( rtGeometryInstanceDestroy( m_geometryinstance ) );
02390   }
02391
02392   inline void GeometryInstanceObj::validate()
02393   {
02394     checkError( rtGeometryInstanceValidate( m_geometryinstance ) );
02395   }
02396
02397   inline Context GeometryInstanceObj::getContext()
02398   {
02399     RTcontext c;
02400     checkErrorNoGetContext( rtGeometryInstanceGetContext( m_geometryinstance, &c
      ) );
02401     return Context::take( c );
02402   }
02403
02404   inline void GeometryInstanceObj::setGeometry(Geometry geometry)
02405   {
02406     checkError( rtGeometryInstanceSetGeometry( m_geometryinstance, geometry->get(
      ) ) );
02407   }
02408
02409   inline Geometry GeometryInstanceObj::getGeometry()
02410   {
02411     RTgeometry result;
02412     checkError( rtGeometryInstanceGetGeometry( m_geometryinstance, &result ) );
02413     return Geometry::take( result );
02414   }
02415
02416   inline void GeometryInstanceObj::setMaterialCount(unsigned int  count)
02417   {
```

```
02418     checkError( rtGeometryInstanceSetMaterialCount( m_geometryinstance, count ) )
      ;
02419   }
02420
02421   inline unsigned int GeometryInstanceObj::getMaterialCount()
02422   {
02423     unsigned int result;
02424     checkError( rtGeometryInstanceGetMaterialCount( m_geometryinstance, &result )
       );
02425     return result;
02426   }
02427
02428   inline void GeometryInstanceObj::setMaterial(unsigned int idx, Material  materi
      al)
02429   {
02430     checkError( rtGeometryInstanceSetMaterial( m_geometryinstance, idx, material-
      >get()) );
02431   }
02432
02433   inline Material GeometryInstanceObj::getMaterial(unsigned int idx)
02434   {
02435     RTmaterial result;
02436     checkError( rtGeometryInstanceGetMaterial( m_geometryinstance, idx, &result )
       );
02437     return Material::take( result );
02438   }
02439
02440   // Adds the material and returns the index to the added material.
02441   inline unsigned int GeometryInstanceObj::addMaterial(Material material)
02442   {
02443     unsigned int old_count = getMaterialCount();
02444     setMaterialCount(old_count+1);
02445     setMaterial(old_count, material);
02446     return old_count;
02447   }
02448
02449   inline Variable GeometryInstanceObj::declareVariable(const std::string& name)
02450   {
02451     RTvariable v;
02452     checkError( rtGeometryInstanceDeclareVariable( m_geometryinstance, name.c_str
      (), &v ) );
02453     return Variable::take( v );
02454   }
02455
02456   inline Variable GeometryInstanceObj::queryVariable(const std::string& name)
02457   {
02458     RTvariable v;
02459     checkError( rtGeometryInstanceQueryVariable( m_geometryinstance, name.c_str()
      , &v ) );
02460     return Variable::take( v );
02461   }
02462
02463   inline void GeometryInstanceObj::removeVariable(Variable v)
02464   {
02465     checkError( rtGeometryInstanceRemoveVariable( m_geometryinstance, v->get() )
      );
02466   }
02467
02468   inline unsigned int GeometryInstanceObj::getVariableCount()
02469   {
02470     unsigned int result;
02471     checkError( rtGeometryInstanceGetVariableCount( m_geometryinstance, &result )
       );
02472     return result;
02473   }
02474
02475   inline Variable GeometryInstanceObj::getVariable(unsigned int index)
```

```
02476   {
02477     RTvariable v;
02478     checkError( rtGeometryInstanceGetVariable( m_geometryinstance, index, &v ) );
02479     return Variable::take( v );
02480   }
02481
02482   inline RTgeometryinstance GeometryInstanceObj::get()
02483   {
02484     return m_geometryinstance;
02485   }
02486
02487   inline void GeometryObj::destroy()
02488   {
02489     checkError( rtGeometryDestroy( m_geometry ) );
02490   }
02491
02492   inline void GeometryObj::validate()
02493   {
02494     checkError( rtGeometryValidate( m_geometry ) );
02495   }
02496
02497   inline Context GeometryObj::getContext()
02498   {
02499     RTcontext c;
02500     checkErrorNoGetContext( rtGeometryGetContext( m_geometry, &c ) );
02501     return Context::take( c );
02502   }
02503
02504   inline void GeometryObj::setPrimitiveCount(unsigned int  num_primitives)
02505   {
02506     checkError( rtGeometrySetPrimitiveCount( m_geometry, num_primitives ) );
02507   }
02508
02509   inline unsigned int GeometryObj::getPrimitiveCount()
02510   {
02511     unsigned int result;
02512     checkError( rtGeometryGetPrimitiveCount( m_geometry, &result ) );
02513     return result;
02514   }
02515
02516   inline void GeometryObj::setBoundingBoxProgram(Program  program)
02517   {
02518     checkError( rtGeometrySetBoundingBoxProgram( m_geometry, program->get() ) );
02519   }
02520
02521   inline Program GeometryObj::getBoundingBoxProgram()
02522   {
02523     RTprogram result;
02524     checkError( rtGeometryGetBoundingBoxProgram( m_geometry, &result ) );
02525     return Program::take( result );
02526   }
02527
02528   inline void GeometryObj::setIntersectionProgram(Program  program)
02529   {
02530     checkError( rtGeometrySetIntersectionProgram( m_geometry, program->get() ) );
02531   }
02532
02533   inline Program GeometryObj::getIntersectionProgram()
02534   {
02535     RTprogram result;
02536     checkError( rtGeometryGetIntersectionProgram( m_geometry, &result ) );
02537     return Program::take( result );
02538   }
02539
02540   inline Variable GeometryObj::declareVariable(const std::string& name)
```

```
02541   {
02542     RTvariable v;
02543     checkError( rtGeometryDeclareVariable( m_geometry, name.c_str(), &v ) );
02544     return Variable::take( v );
02545   }
02546
02547   inline Variable GeometryObj::queryVariable(const std::string& name)
02548   {
02549     RTvariable v;
02550     checkError( rtGeometryQueryVariable( m_geometry, name.c_str(), &v ) );
02551     return Variable::take( v );
02552   }
02553
02554   inline void GeometryObj::removeVariable(Variable v)
02555   {
02556     checkError( rtGeometryRemoveVariable( m_geometry, v->get() ) );
02557   }
02558
02559   inline unsigned int GeometryObj::getVariableCount()
02560   {
02561     unsigned int result;
02562     checkError( rtGeometryGetVariableCount( m_geometry, &result ) );
02563     return result;
02564   }
02565
02566   inline Variable GeometryObj::getVariable(unsigned int index)
02567   {
02568     RTvariable v;
02569     checkError( rtGeometryGetVariable( m_geometry, index, &v ) );
02570     return Variable::take( v );
02571   }
02572
02573   inline void GeometryObj::markDirty()
02574   {
02575     checkError( rtGeometryMarkDirty(m_geometry) );
02576   }
02577
02578   inline bool GeometryObj::isDirty()
02579   {
02580     int dirty;
02581     checkError( rtGeometryIsDirty(m_geometry,&dirty) );
02582     return dirty != 0;
02583   }
02584
02585   inline RTgeometry GeometryObj::get()
02586   {
02587     return m_geometry;
02588   }
02589
02590   inline void MaterialObj::destroy()
02591   {
02592     checkError( rtMaterialDestroy( m_material ) );
02593   }
02594
02595   inline void MaterialObj::validate()
02596   {
02597     checkError( rtMaterialValidate( m_material ) );
02598   }
02599
02600   inline Context MaterialObj::getContext()
02601   {
02602     RTcontext c;
02603     checkErrorNoGetContext( rtMaterialGetContext( m_material, &c ) );
02604     return Context::take( c );
02605   }
02606
02607   inline void MaterialObj::setClosestHitProgram(unsigned int ray_type_index,
```

```
      Program  program)
02608   {
02609     checkError( rtMaterialSetClosestHitProgram( m_material, ray_type_index, progr
      am->get() ) );
02610   }
02611
02612   inline Program MaterialObj::getClosestHitProgram(unsigned int ray_type_index)
02613   {
02614     RTprogram result;
02615     checkError( rtMaterialGetClosestHitProgram( m_material, ray_type_index, &resu
      lt ) );
02616     return Program::take( result );
02617   }
02618
02619   inline void MaterialObj::setAnyHitProgram(unsigned int ray_type_index, Program
       program)
02620   {
02621     checkError( rtMaterialSetAnyHitProgram( m_material, ray_type_index, program->
      get() ) );
02622   }
02623
02624   inline Program MaterialObj::getAnyHitProgram(unsigned int ray_type_index)
02625   {
02626     RTprogram result;
02627     checkError( rtMaterialGetAnyHitProgram( m_material, ray_type_index, &result )
       );
02628     return Program::take( result );
02629   }
02630
02631   inline Variable MaterialObj::declareVariable(const std::string& name)
02632   {
02633     RTvariable v;
02634     checkError( rtMaterialDeclareVariable( m_material, name.c_str(), &v ) );
02635     return Variable::take( v );
02636   }
02637
02638   inline Variable MaterialObj::queryVariable(const std::string& name)
02639   {
02640     RTvariable v;
02641     checkError( rtMaterialQueryVariable( m_material, name.c_str(), &v) );
02642     return Variable::take( v );
02643   }
02644
02645   inline void MaterialObj::removeVariable(Variable v)
02646   {
02647     checkError( rtMaterialRemoveVariable( m_material, v->get() ) );
02648   }
02649
02650   inline unsigned int MaterialObj::getVariableCount()
02651   {
02652     unsigned int result;
02653     checkError( rtMaterialGetVariableCount( m_material, &result ) );
02654     return result;
02655   }
02656
02657   inline Variable MaterialObj::getVariable(unsigned int index)
02658   {
02659     RTvariable v;
02660     checkError( rtMaterialGetVariable( m_material, index, &v) );
02661     return Variable::take( v );
02662   }
02663
02664   inline RTmaterial MaterialObj::get()
02665   {
02666     return m_material;
02667   }
02668
```

```
02669   inline void TextureSamplerObj::destroy()
02670   {
02671     checkError( rtTextureSamplerDestroy( m_texturesampler ) );
02672   }
02673
02674   inline void TextureSamplerObj::validate()
02675   {
02676     checkError( rtTextureSamplerValidate( m_texturesampler ) );
02677   }
02678
02679   inline Context TextureSamplerObj::getContext()
02680   {
02681     RTcontext c;
02682     checkErrorNoGetContext( rtTextureSamplerGetContext( m_texturesampler, &c ) );

02683     return Context::take( c );
02684   }
02685
02686   inline void TextureSamplerObj::setMipLevelCount(unsigned int  num_mip_levels)
02687   {
02688     checkError( rtTextureSamplerSetMipLevelCount(m_texturesampler, num_mip_levels
       ) );
02689   }
02690
02691   inline unsigned int TextureSamplerObj::getMipLevelCount()
02692   {
02693     unsigned int result;
02694     checkError( rtTextureSamplerGetMipLevelCount( m_texturesampler, &result ) );
02695     return result;
02696   }
02697
02698   inline void TextureSamplerObj::setArraySize(unsigned int  num_textures_in_array
      )
02699   {
02700     checkError( rtTextureSamplerSetArraySize( m_texturesampler, num_textures_in_a
      rray ) );
02701   }
02702
02703   inline unsigned int TextureSamplerObj::getArraySize()
02704   {
02705     unsigned int result;
02706     checkError( rtTextureSamplerGetArraySize( m_texturesampler, &result ) );
02707     return result;
02708   }
02709
02710   inline void TextureSamplerObj::setWrapMode(unsigned int dim, RTwrapmode wrapmod
      e)
02711   {
02712     checkError( rtTextureSamplerSetWrapMode( m_texturesampler, dim, wrapmode ) );

02713   }
02714
02715   inline RTwrapmode TextureSamplerObj::getWrapMode(unsigned int dim)
02716   {
02717     RTwrapmode wrapmode;
02718     checkError( rtTextureSamplerGetWrapMode( m_texturesampler, dim, &wrapmode ) )
      ;
02719     return wrapmode;
02720   }
02721
02722   inline void TextureSamplerObj::setFilteringModes(RTfiltermode  minification, RT
      filtermode  magnification, RTfiltermode  mipmapping)
02723   {
02724     checkError( rtTextureSamplerSetFilteringModes( m_texturesampler, minification
      , magnification, mipmapping ) );
02725   }
02726
```

```
02727   inline void TextureSamplerObj::getFilteringModes(RTfiltermode& minification, RT
      filtermode& magnification, RTfiltermode& mipmapping)
02728   {
02729     checkError( rtTextureSamplerGetFilteringModes( m_texturesampler, &minificatio
      n, &magnification, &mipmapping ) );
02730   }
02731
02732   inline void TextureSamplerObj::setMaxAnisotropy(float value)
02733   {
02734     checkError( rtTextureSamplerSetMaxAnisotropy(m_texturesampler, value ) );
02735   }
02736
02737   inline float TextureSamplerObj::getMaxAnisotropy()
02738   {
02739     float result;
02740     checkError( rtTextureSamplerGetMaxAnisotropy( m_texturesampler, &result) );
02741     return result;
02742   }
02743
02744   inline void TextureSamplerObj::setReadMode(RTtexturereadmode  readmode)
02745   {
02746     checkError( rtTextureSamplerSetReadMode( m_texturesampler, readmode ) );
02747   }
02748
02749   inline RTtexturereadmode TextureSamplerObj::getReadMode()
02750   {
02751     RTtexturereadmode result;
02752     checkError( rtTextureSamplerGetReadMode( m_texturesampler, &result) );
02753     return result;
02754   }
02755
02756   inline void TextureSamplerObj::setIndexingMode(RTtextureindexmode  indexmode)
02757   {
02758     checkError( rtTextureSamplerSetIndexingMode( m_texturesampler, indexmode ) );

02759   }
02760
02761   inline RTtextureindexmode TextureSamplerObj::getIndexingMode()
02762   {
02763     RTtextureindexmode result;
02764     checkError( rtTextureSamplerGetIndexingMode( m_texturesampler, &result ) );
02765     return result;
02766   }
02767
02768   inline void TextureSamplerObj::setBuffer(unsigned int texture_array_idx, unsign
      ed int mip_level, Buffer buffer)
02769   {
02770     checkError( rtTextureSamplerSetBuffer( m_texturesampler, texture_array_idx, m
      ip_level, buffer->get() ) );
02771   }
02772
02773   inline Buffer TextureSamplerObj::getBuffer(unsigned int texture_array_idx, unsi
      gned int mip_level)
02774   {
02775     RTbuffer result;
02776     checkError( rtTextureSamplerGetBuffer(m_texturesampler, texture_array_idx, mi
      p_level, &result ) );
02777     return Buffer::take(result);
02778   }
02779
02780   inline RTtexturesampler TextureSamplerObj::get()
02781   {
02782     return m_texturesampler;
02783   }
02784
02785   inline void TextureSamplerObj::registerGLTexture()
02786   {
```

```
02787     checkError( rtTextureSamplerGLRegister( m_texturesampler ) );
02788   }
02789
02790   inline void TextureSamplerObj::unregisterGLTexture()
02791   {
02792     checkError( rtTextureSamplerGLUnregister( m_texturesampler ) );
02793   }
02794
02795 #ifdef _WIN32
02796
02797   inline void TextureSamplerObj::registerD3D9Texture()
02798   {
02799     checkError( rtTextureSamplerD3D9Register( m_texturesampler ) );
02800   }
02801
02802   inline void TextureSamplerObj::registerD3D10Texture()
02803   {
02804     checkError( rtTextureSamplerD3D10Register( m_texturesampler ) );
02805   }
02806
02807   inline void TextureSamplerObj::registerD3D11Texture()
02808   {
02809     checkError( rtTextureSamplerD3D11Register( m_texturesampler ) );
02810   }
02811
02812   inline void TextureSamplerObj::unregisterD3D9Texture()
02813   {
02814     checkError( rtTextureSamplerD3D9Unregister( m_texturesampler ) );
02815   }
02816
02817   inline void TextureSamplerObj::unregisterD3D10Texture()
02818   {
02819     checkError( rtTextureSamplerD3D10Unregister( m_texturesampler ) );
02820   }
02821
02822   inline void TextureSamplerObj::unregisterD3D11Texture()
02823   {
02824     checkError( rtTextureSamplerD3D11Unregister( m_texturesampler ) );
02825   }
02826
02827 #endif
02828
02829   inline void BufferObj::destroy()
02830   {
02831     checkError( rtBufferDestroy( m_buffer ) );
02832   }
02833
02834   inline void BufferObj::validate()
02835   {
02836     checkError( rtBufferValidate( m_buffer ) );
02837   }
02838
02839   inline Context BufferObj::getContext()
02840   {
02841     RTcontext c;
02842     checkErrorNoGetContext( rtBufferGetContext( m_buffer, &c ) );
02843     return Context::take( c );
02844   }
02845
02846   inline void BufferObj::setFormat(RTformat format)
02847   {
02848     checkError( rtBufferSetFormat( m_buffer, format ) );
02849   }
02850
02851   inline RTformat BufferObj::getFormat()
02852   {
02853     RTformat result;
```

```
02854     checkError( rtBufferGetFormat( m_buffer, &result ) );
02855     return result;
02856   }
02857
02858   inline void BufferObj::setElementSize(RTsize size_of_element)
02859   {
02860     checkError( rtBufferSetElementSize ( m_buffer, size_of_element ) );
02861   }
02862
02863   inline RTsize BufferObj::getElementSize()
02864   {
02865     RTsize result;
02866     checkError( rtBufferGetElementSize ( m_buffer, &result) );
02867     return result;
02868   }
02869
02870   inline void BufferObj::setSize(RTsize width)
02871   {
02872     checkError( rtBufferSetSize1D( m_buffer, width ) );
02873   }
02874
02875   inline void BufferObj::getSize(RTsize& width)
02876   {
02877     checkError( rtBufferGetSize1D( m_buffer, &width ) );
02878   }
02879
02880   inline void BufferObj::setSize(RTsize width, RTsize height)
02881   {
02882     checkError( rtBufferSetSize2D( m_buffer, width, height ) );
02883   }
02884
02885   inline void BufferObj::getSize(RTsize& width, RTsize& height)
02886   {
02887     checkError( rtBufferGetSize2D( m_buffer, &width, &height ) );
02888   }
02889
02890   inline void BufferObj::setSize(RTsize width, RTsize height, RTsize depth)
02891   {
02892     checkError( rtBufferSetSize3D( m_buffer, width, height, depth ) );
02893   }
02894
02895   inline void BufferObj::getSize(RTsize& width, RTsize& height, RTsize& depth)
02896   {
02897     checkError( rtBufferGetSize3D( m_buffer, &width, &height, &depth ) );
02898   }
02899
02900   inline void BufferObj::setSize(unsigned int dimensionality, const RTsize* dims)

02901   {
02902     checkError( rtBufferSetSizev( m_buffer, dimensionality, dims ) );
02903   }
02904
02905   inline void BufferObj::getSize(unsigned int dimensionality, RTsize* dims)
02906   {
02907     checkError( rtBufferGetSizev( m_buffer, dimensionality, dims ) );
02908   }
02909
02910   inline unsigned int BufferObj::getDimensionality()
02911   {
02912     unsigned int result;
02913     checkError( rtBufferGetDimensionality( m_buffer, &result ) );
02914     return result;
02915   }
02916
02917   inline unsigned int BufferObj::getGLBOId()
02918   {
02919     unsigned int result;
```

```
02920     checkError( rtBufferGetGLBOId( m_buffer, &result ) );
02921     return result;
02922   }
02923
02924   inline void BufferObj::registerGLBuffer()
02925   {
02926     checkError( rtBufferGLRegister( m_buffer ) );
02927   }
02928
02929   inline void BufferObj::unregisterGLBuffer()
02930   {
02931     checkError( rtBufferGLUnregister( m_buffer ) );
02932   }
02933
02934 #ifdef _WIN32
02935
02936   inline void BufferObj::registerD3D9Buffer()
02937   {
02938     checkError( rtBufferD3D9Register( m_buffer ) );
02939   }
02940
02941   inline void BufferObj::registerD3D10Buffer()
02942   {
02943     checkError( rtBufferD3D10Register( m_buffer ) );
02944   }
02945
02946   inline void BufferObj::registerD3D11Buffer()
02947   {
02948     checkError( rtBufferD3D11Register( m_buffer ) );
02949   }
02950
02951   inline void BufferObj::unregisterD3D9Buffer()
02952   {
02953     checkError( rtBufferD3D9Unregister( m_buffer ) );
02954   }
02955
02956   inline void BufferObj::unregisterD3D10Buffer()
02957   {
02958     checkError( rtBufferD3D10Unregister( m_buffer ) );
02959   }
02960
02961   inline void BufferObj::unregisterD3D11Buffer()
02962   {
02963     checkError( rtBufferD3D11Unregister( m_buffer ) );
02964   }
02965
02966   inline IDirect3DResource9* BufferObj::getD3D9Resource()
02967   {
02968     IDirect3DResource9* result = NULL;
02969     checkError( rtBufferGetD3D9Resource( m_buffer, &result ) );
02970     return result;
02971   }
02972
02973   inline ID3D10Resource* BufferObj::getD3D10Resource()
02974   {
02975     ID3D10Resource* result = NULL;
02976     checkError( rtBufferGetD3D10Resource( m_buffer, &result ) );
02977     return result;
02978   }
02979
02980   inline ID3D11Resource* BufferObj::getD3D11Resource()
02981   {
02982     ID3D11Resource* result = NULL;
02983     checkError( rtBufferGetD3D11Resource( m_buffer, &result ) );
02984     return result;
02985   }
02986
```

```
02987 #endif
02988
02989   inline void* BufferObj::map()
02990   {
02991     void* result;
02992     checkError( rtBufferMap( m_buffer, &result ) );
02993     return result;
02994   }
02995
02996   inline void BufferObj::unmap()
02997   {
02998     checkError( rtBufferUnmap( m_buffer ) );
02999   }
03000
03001
03002   inline RTbuffer BufferObj::get()
03003   {
03004     return m_buffer;
03005   }
03006
03007   inline Context VariableObj::getContext()
03008   {
03009     RTcontext c;
03010     checkErrorNoGetContext( rtVariableGetContext( m_variable, &c ) );
03011     return Context::take( c );
03012   }
03013
03014   inline void VariableObj::setUint(unsigned int u1)
03015   {
03016     checkError( rtVariableSet1ui( m_variable, u1 ) );
03017   }
03018
03019   inline void VariableObj::setUint(unsigned int u1, unsigned int u2)
03020   {
03021     checkError( rtVariableSet2ui( m_variable, u1, u2 ) );
03022   }
03023
03024   inline void VariableObj::setUint(unsigned int u1, unsigned int u2, unsigned int
      u3)
03025   {
03026     checkError( rtVariableSet3ui( m_variable, u1, u2, u3 ) );
03027   }
03028
03029   inline void VariableObj::setUint(unsigned int u1, unsigned int u2, unsigned int
      u3, unsigned int u4)
03030   {
03031     checkError( rtVariableSet4ui( m_variable, u1, u2, u3, u4 ) );
03032   }
03033
03034   inline void VariableObj::set1uiv(const unsigned int* u)
03035   {
03036     checkError( rtVariableSet1uiv( m_variable, u ) );
03037   }
03038
03039   inline void VariableObj::set2uiv(const unsigned int* u)
03040   {
03041     checkError( rtVariableSet2uiv( m_variable, u ) );
03042   }
03043
03044   inline void VariableObj::set3uiv(const unsigned int* u)
03045   {
03046     checkError( rtVariableSet3uiv( m_variable, u ) );
03047   }
03048
03049   inline void VariableObj::set4uiv(const unsigned int* u)
03050   {
03051     checkError( rtVariableSet4uiv( m_variable, u ) );
```

```
03052   }
03053
03054   inline void VariableObj::setMatrix2x2fv(bool transpose, const float* m)
03055   {
03056     checkError( rtVariableSetMatrix2x2fv( m_variable, (int)transpose, m ) );
03057   }
03058
03059   inline void VariableObj::setMatrix2x3fv(bool transpose, const float* m)
03060   {
03061     checkError( rtVariableSetMatrix2x3fv( m_variable, (int)transpose, m ) );
03062   }
03063
03064   inline void VariableObj::setMatrix2x4fv(bool transpose, const float* m)
03065   {
03066     checkError( rtVariableSetMatrix2x4fv( m_variable, (int)transpose, m ) );
03067   }
03068
03069   inline void VariableObj::setMatrix3x2fv(bool transpose, const float* m)
03070   {
03071     checkError( rtVariableSetMatrix3x2fv( m_variable, (int)transpose, m ) );
03072   }
03073
03074   inline void VariableObj::setMatrix3x3fv(bool transpose, const float* m)
03075   {
03076     checkError( rtVariableSetMatrix3x3fv( m_variable, (int)transpose, m ) );
03077   }
03078
03079   inline void VariableObj::setMatrix3x4fv(bool transpose, const float* m)
03080   {
03081     checkError( rtVariableSetMatrix3x4fv( m_variable, (int)transpose, m ) );
03082   }
03083
03084   inline void VariableObj::setMatrix4x2fv(bool transpose, const float* m)
03085   {
03086     checkError( rtVariableSetMatrix4x2fv( m_variable, (int)transpose, m ) );
03087   }
03088
03089   inline void VariableObj::setMatrix4x3fv(bool transpose, const float* m)
03090   {
03091     checkError( rtVariableSetMatrix4x3fv( m_variable, (int)transpose, m ) );
03092   }
03093
03094   inline void VariableObj::setMatrix4x4fv(bool transpose, const float* m)
03095   {
03096     checkError( rtVariableSetMatrix4x4fv( m_variable, (int)transpose, m ) );
03097   }
03098
03099   inline void VariableObj::setFloat(float f1)
03100   {
03101     checkError( rtVariableSet1f( m_variable, f1 ) );
03102   }
03103
03104   inline void VariableObj::setFloat(optix::float2 f)
03105   {
03106     checkError( rtVariableSet2fv( m_variable, &f.x ) );
03107   }
03108
03109   inline void VariableObj::setFloat(float f1, float f2)
03110   {
03111     checkError( rtVariableSet2f( m_variable, f1, f2 ) );
03112   }
03113
03114   inline void VariableObj::setFloat(optix::float3 f)
03115   {
03116     checkError( rtVariableSet3fv( m_variable, &f.x ) );
03117   }
03118
```

```
03119   inline void VariableObj::setFloat(float f1, float f2, float f3)
03120   {
03121     checkError( rtVariableSet3f( m_variable, f1, f2, f3 ) );
03122   }
03123
03124   inline void VariableObj::setFloat(optix::float4 f)
03125   {
03126     checkError( rtVariableSet4fv( m_variable, &f.x ) );
03127   }
03128
03129   inline void VariableObj::setFloat(float f1, float f2, float f3, float f4)
03130   {
03131     checkError( rtVariableSet4f( m_variable, f1, f2, f3, f4 ) );
03132   }
03133
03134   inline void VariableObj::set1fv(const float* f)
03135   {
03136     checkError( rtVariableSet1fv( m_variable, f ) );
03137   }
03138
03139   inline void VariableObj::set2fv(const float* f)
03140   {
03141     checkError( rtVariableSet2fv( m_variable, f ) );
03142   }
03143
03144   inline void VariableObj::set3fv(const float* f)
03145   {
03146     checkError( rtVariableSet3fv( m_variable, f ) );
03147   }
03148
03149   inline void VariableObj::set4fv(const float* f)
03150   {
03151     checkError( rtVariableSet4fv( m_variable, f ) );
03152   }
03153
03155   inline void VariableObj::setInt(int i1)
03156   {
03157     checkError( rtVariableSet1i( m_variable, i1 ) );
03158   }
03159
03160   inline void VariableObj::setInt(optix::int2 i)
03161   {
03162     checkError( rtVariableSet2iv( m_variable, &i.x ) );
03163   }
03164
03165   inline void VariableObj::setInt(int i1, int i2)
03166   {
03167     checkError( rtVariableSet2i( m_variable, i1, i2 ) );
03168   }
03169
03170   inline void VariableObj::setInt(optix::int3 i)
03171   {
03172     checkError( rtVariableSet3iv( m_variable, &i.x ) );
03173   }
03174
03175   inline void VariableObj::setInt(int i1, int i2, int i3)
03176   {
03177     checkError( rtVariableSet3i( m_variable, i1, i2, i3 ) );
03178   }
03179
03180   inline void VariableObj::setInt(optix::int4 i)
03181   {
03182     checkError( rtVariableSet4iv( m_variable, &i.x ) );
03183   }
03184
03185   inline void VariableObj::setInt(int i1, int i2, int i3, int i4)
03186   {
```

```
03187     checkError( rtVariableSet4i( m_variable, i1, i2, i3, i4 ) );
03188   }
03189
03190   inline void VariableObj::set1iv( const int* i )
03191   {
03192     checkError( rtVariableSet1iv( m_variable, i ) );
03193   }
03194
03195   inline void VariableObj::set2iv( const int* i )
03196   {
03197     checkError( rtVariableSet2iv( m_variable, i ) );
03198   }
03199
03200   inline void VariableObj::set3iv( const int* i )
03201   {
03202     checkError( rtVariableSet3iv( m_variable, i ) );
03203   }
03204
03205   inline void VariableObj::set4iv( const int* i )
03206   {
03207     checkError( rtVariableSet4iv( m_variable, i ) );
03208   }
03209
03210   inline float VariableObj::getFloat()
03211   {
03212     float f;
03213     checkError( rtVariableGet1f( m_variable, &f ) );
03214     return f;
03215   }
03216
03217   inline unsigned int VariableObj::getUint()
03218   {
03219     unsigned int i;
03220     checkError( rtVariableGet1ui( m_variable, &i ) );
03221     return i;
03222   }
03223
03224   inline int VariableObj::getInt()
03225   {
03226     int i;
03227     checkError( rtVariableGet1i( m_variable, &i ) );
03228     return i;
03229   }
03230
03231   inline void VariableObj::setBuffer(Buffer buffer)
03232   {
03233     checkError( rtVariableSetObject( m_variable, buffer->get() ) );
03234   }
03235
03236   inline void VariableObj::set(Buffer buffer)
03237   {
03238     checkError( rtVariableSetObject( m_variable, buffer->get() ) );
03239   }
03240
03241   inline void VariableObj::setUserData(RTsize size, const void* ptr)
03242   {
03243     checkError( rtVariableSetUserData( m_variable, size, ptr ) );
03244   }
03245
03246   inline void VariableObj::getUserData(RTsize size,       void* ptr)
03247   {
03248     checkError( rtVariableGetUserData( m_variable, size, ptr ) );
03249   }
03250
03251   inline void VariableObj::setTextureSampler(TextureSampler texturesampler)
03252   {
03253     checkError( rtVariableSetObject( m_variable, texturesampler->get() ) );
```

```
03254   }
03255
03256   inline void VariableObj::set(TextureSampler texturesampler)
03257   {
03258     checkError( rtVariableSetObject( m_variable, texturesampler->get() ) );
03259   }
03260
03261   inline void VariableObj::set(GeometryGroup group)
03262   {
03263     checkError( rtVariableSetObject( m_variable, group->get() ) );
03264   }
03265
03266   inline void VariableObj::set(Group group)
03267   {
03268     checkError( rtVariableSetObject( m_variable, group->get() ) );
03269   }
03270
03271   inline void VariableObj::set(Selector sel)
03272   {
03273     checkError( rtVariableSetObject( m_variable, sel->get() ) );
03274   }
03275
03276   inline void VariableObj::set(Transform tran)
03277   {
03278     checkError( rtVariableSetObject( m_variable, tran->get() ) );
03279   }
03280
03281   inline Buffer VariableObj::getBuffer()
03282   {
03283     RTobject temp;
03284     checkError( rtVariableGetObject( m_variable, &temp ) );
03285     RTbuffer buffer = reinterpret_cast<RTbuffer>(temp);
03286     return Buffer::take(buffer);
03287   }
03288
03289   inline std::string VariableObj::getName()
03290   {
03291     const char* name;
03292     checkError( rtVariableGetName( m_variable, &name ) );
03293     return std::string(name);
03294   }
03295
03296   inline std::string VariableObj::getAnnotation()
03297   {
03298     const char* annotation;
03299     checkError( rtVariableGetAnnotation( m_variable, &annotation ) );
03300     return std::string(annotation);
03301   }
03302
03303   inline RTobjecttype VariableObj::getType()
03304   {
03305     RTobjecttype type;
03306     checkError( rtVariableGetType( m_variable, &type ) );
03307     return type;
03308   }
03309
03310   inline RTvariable VariableObj::get()
03311   {
03312     return m_variable;
03313   }
03314
03315   inline RTsize VariableObj::getSize()
03316   {
03317     RTsize size;
03318     checkError( rtVariableGetSize( m_variable, &size ) );
03319     return size;
03320   }
```

```
03321
03322   inline optix::TextureSampler VariableObj::getTextureSampler()
03323   {
03324     RTobject temp;
03325     checkError( rtVariableGetObject( m_variable, &temp ) );
03326     RTtexturesampler sampler = reinterpret_cast<RTtexturesampler>(temp);
03327     return TextureSampler::take(sampler);
03328   }
03329
03331 }
03332
03333 #endif /* __optixu_optixpp_namespace_h__ */
03334
03336
```

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, RTgeometryinstance 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)

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 rtuCUDACompileFile (const char * *filename*, const char ** *preprocessorArguments*, unsigned int *numPreprocessorArguments*, RTsize * *resultSize*, RTsize * *errorSize*)

3.3.2.2 RTresult RTAPI rtuCUDACompileString (const char * *source*, const char ** *preprocessorArguments*, unsigned int *numPreprocessorArguments*, RTsize * *resultSize*, RTsize * *errorSize*)

3.3.2.3 RTresult RTAPI rtuCUDAGetCompileResult (char * *result*, char * *error*)

3.3.2.4 RTU_INLINE RTresult rtuGeometryGroupAddChild (RTgeometrygroup *geometrygroup*, RTgeometryinstance *child*, unsigned int * *index*)

Definition at line 273 of file optixu.h.

3.3.2.5 RTU_INLINE RTresult rtuGeometryGroupGetChildIndex (RTgeometrygroup *geometrygroup*, RTgeometryinstance *child*, unsigned int * *index*)

Definition at line 366 of file optixu.h.

3.3.2.6 RTU_INLINE RTresult rtuGeometryGroupRemoveChild (RTgeometrygroup *geometrygroup*, RTgeometryinstance *child*)

Definition at line 299 of file optixu.h.

3.3.2.7 RTU_INLINE RTresult rtuGeometryGroupRemoveChildByIndex (RTgeometrygroup *geometrygroup*, unsigned int *index*)

Definition at line 329 of file optixu.h.

3.3.2.8 RTresult RTAPI rtuGetSizeForRTformat (RTformat *format*, size_t * *size*)

3.3.2.9 RTU_INLINE RTresult rtuGroupAddChild (RTgroup *group*, RTobject *child*, unsigned int * *index*)

Definition at line 180 of file optixu.h.

3.3.2.10 RTU_INLINE RTresult rtuGroupGetChildIndex (RTgroup *group*, RTobject *child*, unsigned int * *index*)

Definition at line 340 of file optixu.h.

3.3.2.11 RTU_INLINE RTresult rtuGroupRemoveChild (RTgroup *group*, RTobject *child*)

Definition at line 283 of file optixu.h.

3.3.2.12 RTU_INLINE RTresult rtuGroupRemoveChildByIndex (RTgroup *group*, unsigned int *index*)

Definition at line 307 of file optixu.h.

3.3.2.13 RTresult RTAPI rtuNameForType (RTobjecttype *type*, char * *buffer*, RTsize *bufferSize*)

3.3.2.14 RTU_INLINE RTresult rtuSelectorAddChild (RTselector *selector*, RTobject *child*, unsigned int * *index*)

Definition at line 185 of file optixu.h.

3.3.2.15 RTU_INLINE RTresult rtuSelectorGetChildIndex (RTselector *selector*, RTobject *child*, unsigned int * *index*)

Definition at line 353 of file optixu.h.

3.3.2.16 RTU_INLINE RTresult rtuSelectorRemoveChild (RTselector *selector*, RTobject *child*)

Definition at line 291 of file optixu.h.

3.3.2.17 RTU_INLINE RTresult rtuSelectorRemoveChildByIndex (RTselector *selector*, unsigned int *index*)

Definition at line 318 of file optixu.h.

3.3.2.18 RTU_INLINE RTresult rtuTransformSetChild (RTtransform *transform*, RTobject *child*)

Definition at line 239 of file optixu.h.

3.4 optixu.h

```
00001
00002 /*
00003  * Copyright (c) 2008 - 2009 NVIDIA Corporation.  All rights reserved.
00004  *
00005  * NVIDIA Corporation and its licensors retain all intellectual property and prop
      rietary
00006  * rights in and to this software, related documentation and any modifications th
      ereto.
00007  * Any use, reproduction, disclosure or distribution of this software and related

00008  * documentation without an express license agreement from NVIDIA Corporation is
      strictly
00009  * prohibited.
00010  *
00011  * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *
      AS IS*
00012  * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIE
      D,
00013  * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNE
      SS FOR A
00014  * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR A
      NY
00015  * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING,
       WITHOUT
00016  * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS
      OF
00017  * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF O
      R
00018  * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBI
      LITY OF
00019  * SUCH DAMAGES
00020  */
00021
00022 #ifndef __optix_optixu_h__
00023 #define __optix_optixu_h__
00024
00025 #include <stddef.h>
00026 #include <optix.h>
00027
00028 #ifdef __cplusplus
00029 #  define RTU_INLINE inline
00030 #else
00031 #  ifdef _MSC_VER
00032 #    define RTU_INLINE __inline
00033 #  else
00034 #    define RTU_INLINE inline
00035 #  endif
00036 #endif
00037
00038 #ifdef __cplusplus
00039 extern "C" {
00040 #endif
00041
00042  /*
00043   * Get the name string of a given type.
00044   */
00045   RTresult RTAPI rtuNameForType( RTobjecttype type, char* buffer, RTsize bufferSi
      ze );
00046
00047  /*
00048   * Return the size of a given RTformat.  RT_FORMAT_USER and RT_FORMAT_UNKNOWN re
      turn 0.
00049   * Returns RT_ERROR_INVALID_VALUE if the format isn't recognized, RT_SUCCESS oth
      erwise.
00050   */
```

```
00051   RTresult RTAPI rtuGetSizeForRTformat( RTformat format, size_t* size);
00052
00053  /*
00054   * Compile a cuda source string.
00055   * ARGS:
00056   *
00057   * source                       source code string
00058   * preprocessorArguments        list of preprocessor arguments
00059   * numPreprocessorArguments     number of preprocessor arguments
00060   * resultSize                   [out] size required to hold compiled result stri
      ng
00061   * errorSize                    [out] size required to hold error string
00062   */
00063   RTresult RTAPI rtuCUDACompileString( const char* source, const char** preproces
      sorArguments, unsigned int numPreprocessorArguments, RTsize* resultSize, RTsize*
      errorSize );
00064
00065  /*
00066   * Compile a cuda source file.
00067   * ARGS:
00068   *
00069   * filename                     source code file name
00070   * preprocessorArguments        list of preprocessor arguments
00071   * numPreprocessorArguments     number of preprocessor arguments
00072   * resultSize                   [out] size required to hold compiled result stri
      ng
00073   * errorSize                    [out] size required to hold error string
00074   */
00075   RTresult RTAPI rtuCUDACompileFile( const char* filename, const char** preproces
      sorArguments, unsigned int numPreprocessorArguments, RTsize* resultSize, RTsize*
      errorSize );
00076
00077  /*
00078   * Get the result of the most recent call to one of the above compile functions.

00079   * The 'result' and 'error' parameters must point to memory large enough to hold

00080   * the respective strings, as returned by the compile function.
00081   * ARGS:
00082   *
00083   * result                     compiled result string
00084   * error                      error string
00085   */
00086   RTresult RTAPI rtuCUDAGetCompileResult( char* result, char* error );
00087
00088 #ifdef __cplusplus
00089 } /* extern "C" */
00090 #endif
00091
00092  /*
00093   * Add an entry to the end of the child array.
00094   * Fills 'index' with the index of the added child, if the pointer is non-NULL.
00095   */
00096 #ifndef __cplusplus
00097  RTresult rtuGroupAddChild        ( RTgroup group, RTobject child, unsigned int*
      index );
00098  RTresult rtuSelectorAddChild     ( RTselector selector, RTobject child, unsigned
       int* index );
00099 #else
00100  RTresult rtuGroupAddChild        ( RTgroup group, RTgroup         child, unsigne
      d int* index );
00101  RTresult rtuGroupAddChild        ( RTgroup group, RTselector      child, unsigne
      d int* index );
00102  RTresult rtuGroupAddChild        ( RTgroup group, RTtransform     child, unsigne
      d int* index );
00103  RTresult rtuGroupAddChild        ( RTgroup group, RTgeometrygroup child, unsigne
      d int* index );
```

```
00104  RTresult rtuSelectorAddChild     ( RTselector selector, RTgroup         child, u
       nsigned int* index );
00105  RTresult rtuSelectorAddChild     ( RTselector selector, RTselector      child, u
       nsigned int* index );
00106  RTresult rtuSelectorAddChild     ( RTselector selector, RTtransform     child, u
       nsigned int* index );
00107  RTresult rtuSelectorAddChild     ( RTselector selector, RTgeometrygroup child, u
       nsigned int* index );
00108 #endif
00109  RTresult rtuGeometryGroupAddChild( RTgeometrygroup geometrygroup, RTgeometryinst
       ance child, unsigned int* index );
00110
00111  /*
00112   * Wrap rtTransformSetChild in order to provide a type-safe version for C++.
00113   */
00114 #ifndef __cplusplus
00115  RTresult rtuTransformSetChild    ( RTtransform transform, RTobject        child
       );
00116 #else
00117  RTresult rtuTransformSetChild    ( RTtransform transform, RTgroup         child
       );
00118  RTresult rtuTransformSetChild    ( RTtransform transform, RTselector      child
       );
00119  RTresult rtuTransformSetChild    ( RTtransform transform, RTtransform     child
       );
00120  RTresult rtuTransformSetChild    ( RTtransform transform, RTgeometrygroup child
       );
00121 #endif
00122
00123  /*
00124   * Find the given child using a linear search in the child array and remove
00125   * it. If it's not the last entry in the child array, the last entry in the
00126   * array will replace the deleted entry, in order to shrink the array size by on
       e.
00127   */
00128  RTresult rtuGroupRemoveChild        ( RTgroup group, RTobject child );
00129  RTresult rtuSelectorRemoveChild     ( RTselector selector, RTobject child );
00130  RTresult rtuGeometryGroupRemoveChild( RTgeometrygroup geometrygroup, RTgeometryi
       nstance child );
00131
00132  /*
00133   * Remove the child at the given index in the child array. If it's not the last
00134   * entry in the child array, the last entry in the array will replace the delete
       d
00135   * entry, in order to shrink the array size by one.
00136   */
00137  RTU_INLINE RTresult rtuGroupRemoveChildByIndex        ( RTgroup group, unsigned
       int index );
00138  RTU_INLINE RTresult rtuSelectorRemoveChildByIndex     ( RTselector selector, uns
       igned int index );
00139  RTU_INLINE RTresult rtuGeometryGroupRemoveChildByIndex( RTgeometrygroup geometry
       group, unsigned int index );
00140
00141  /*
00142   * Use a linear search to find the child in the child array, and return its inde
       x.
00143   * Returns RT_SUCCESS if the child was found, RT_INVALID_VALUE otherwise.
00144   */
00145  RTU_INLINE RTresult rtuGroupGetChildIndex        ( RTgroup group, RTobject child
       , unsigned int* index );
00146  RTU_INLINE RTresult rtuSelectorGetChildIndex     ( RTselector selector, RTobject
        child, unsigned int* index );
00147  RTU_INLINE RTresult rtuGeometryGroupGetChildIndex( RTgeometrygroup geometrygroup
       , RTgeometryinstance child, unsigned int* index );
00148
00149
00150  /*
```

```
00151   * The following implements the child management helpers declared above.
00152   */
00153
00154 #define RTU_CHECK_ERROR( func )                 \
00155   do {                                          \
00156     RTresult code = func;                       \
00157     if( code != RT_SUCCESS )                    \
00158       return code;                              \
00159   } while(0)
00160
00161 #define RTU_GROUP_ADD_CHILD( _parent, _child, _index )                  \
00162   unsigned int _count;                                                  \
00163   RTU_CHECK_ERROR( rtGroupGetChildCount( (_parent), &_count ) );        \
00164   RTU_CHECK_ERROR( rtGroupSetChildCount( (_parent), _count+1 ) );       \
00165   RTU_CHECK_ERROR( rtGroupSetChild( (_parent), _count, (_child) ) );    \
00166   if( _index ) *(_index) = _count;                                      \
00167   return RT_SUCCESS
00168
00169 #define RTU_SELECTOR_ADD_CHILD( _parent, _child, _index )               \
00170   unsigned int _count;                                                  \
00171   RTU_CHECK_ERROR( rtSelectorGetChildCount( (_parent), &_count ) );     \
00172   RTU_CHECK_ERROR( rtSelectorSetChildCount( (_parent), _count+1 ) );    \
00173   RTU_CHECK_ERROR( rtSelectorSetChild( (_parent), _count, (_child) ) ); \
00174   if( _index ) *(_index) = _count;                                      \
00175   return RT_SUCCESS
00176
00177
00178 #ifndef __cplusplus
00179
00180  RTU_INLINE RTresult rtuGroupAddChild( RTgroup group, RTobject child, unsigned in
      t* index )
00181  {
00182    RTU_GROUP_ADD_CHILD( group, child, index );
00183  }
00184
00185  RTU_INLINE RTresult rtuSelectorAddChild( RTselector selector, RTobject child, un
      signed int* index )
00186  {
00187    RTU_SELECTOR_ADD_CHILD( selector, child, index );
00188  }
00189
00190 #else /* __cplusplus */
00191
00192  RTU_INLINE RTresult rtuGroupAddChild( RTgroup group, RTgroup child, unsigned int
      * index )
00193  {
00194    RTU_GROUP_ADD_CHILD( group, child, index );
00195  }
00196
00197  RTU_INLINE RTresult rtuGroupAddChild( RTgroup group, RTselector child, unsigned
      int* index )
00198  {
00199    RTU_GROUP_ADD_CHILD( group, child, index );
00200  }
00201
00202  RTU_INLINE RTresult rtuGroupAddChild( RTgroup group, RTtransform child, unsigned
       int* index )
00203  {
00204    RTU_GROUP_ADD_CHILD( group, child, index );
00205  }
00206
00207  RTU_INLINE RTresult rtuGroupAddChild( RTgroup group, RTgeometrygroup child, unsi
      gned int* index )
00208  {
00209    RTU_GROUP_ADD_CHILD( group, child, index );
00210  }
00211
```

```
00212  RTU_INLINE RTresult rtuSelectorAddChild( RTselector selector, RTgroup child, uns
       igned int* index )
00213  {
00214    RTU_SELECTOR_ADD_CHILD( selector, child, index );
00215  }
00216
00217  RTU_INLINE RTresult rtuSelectorAddChild( RTselector selector, RTselector child,
       unsigned int* index )
00218  {
00219    RTU_SELECTOR_ADD_CHILD( selector, child, index );
00220  }
00221
00222  RTU_INLINE RTresult rtuSelectorAddChild( RTselector selector, RTtransform child,
       unsigned int* index )
00223  {
00224    RTU_SELECTOR_ADD_CHILD( selector, child, index );
00225  }
00226
00227  RTU_INLINE RTresult rtuSelectorAddChild( RTselector selector, RTgeometrygroup ch
       ild, unsigned int* index )
00228  {
00229    RTU_SELECTOR_ADD_CHILD( selector, child, index );
00230  }
00231
00232 #endif /* __cplusplus */
00233
00234 #undef RTU_GROUP_ADD_CHILD
00235 #undef RTU_SELECTOR_ADD_CHILD
00236
00237 #ifndef __cplusplus
00238
00239  RTU_INLINE RTresult rtuTransformSetChild( RTtransform transform, RTobject child
       )
00240  {
00241    RTU_CHECK_ERROR( rtTransformSetChild( transform, child ) );
00242    return RT_SUCCESS;
00243  }
00244
00245 #else /* __cplusplus */
00246
00247  RTU_INLINE RTresult rtuTransformSetChild( RTtransform transform, RTgroup child )

00248  {
00249    RTU_CHECK_ERROR( rtTransformSetChild( transform, child ) );
00250    return RT_SUCCESS;
00251  }
00252
00253  RTU_INLINE RTresult rtuTransformSetChild( RTtransform transform, RTselector chil
       d )
00254  {
00255    RTU_CHECK_ERROR( rtTransformSetChild( transform, child ) );
00256    return RT_SUCCESS;
00257  }
00258
00259  RTU_INLINE RTresult rtuTransformSetChild( RTtransform transform, RTtransform chi
       ld )
00260  {
00261    RTU_CHECK_ERROR( rtTransformSetChild( transform, child ) );
00262    return RT_SUCCESS;
00263  }
00264
00265  RTU_INLINE RTresult rtuTransformSetChild( RTtransform transform, RTgeometrygroup
        child )
00266  {
00267    RTU_CHECK_ERROR( rtTransformSetChild( transform, child ) );
00268    return RT_SUCCESS;
00269  }
```

```
00270
00271 #endif /* __cplusplus */
00272
00273  RTU_INLINE RTresult rtuGeometryGroupAddChild( RTgeometrygroup geometrygroup, RTg
      eometryinstance child, unsigned int* index )
00274  {
00275    unsigned int count;
00276    RTU_CHECK_ERROR( rtGeometryGroupGetChildCount( geometrygroup, &count ) );
00277    RTU_CHECK_ERROR( rtGeometryGroupSetChildCount( geometrygroup, count+1 ) );
00278    RTU_CHECK_ERROR( rtGeometryGroupSetChild( geometrygroup, count, child ) );
00279    if( index ) *index = count;
00280    return RT_SUCCESS;
00281  }
00282
00283  RTU_INLINE RTresult rtuGroupRemoveChild( RTgroup group, RTobject child )
00284  {
00285    unsigned int index;
00286    RTU_CHECK_ERROR( rtuGroupGetChildIndex( group, child, &index ) );
00287    RTU_CHECK_ERROR( rtuGroupRemoveChildByIndex( group, index ) );
00288    return RT_SUCCESS;
00289  }
00290
00291  RTU_INLINE RTresult rtuSelectorRemoveChild( RTselector selector, RTobject child
      )
00292  {
00293    unsigned int index;
00294    RTU_CHECK_ERROR( rtuSelectorGetChildIndex( selector, child, &index ) );
00295    RTU_CHECK_ERROR( rtuSelectorRemoveChildByIndex( selector, index ) );
00296    return RT_SUCCESS;
00297  }
00298
00299  RTU_INLINE RTresult rtuGeometryGroupRemoveChild( RTgeometrygroup geometrygroup,
      RTgeometryinstance child )
00300  {
00301    unsigned int index;
00302    RTU_CHECK_ERROR( rtuGeometryGroupGetChildIndex( geometrygroup, child, &index )
       );
00303    RTU_CHECK_ERROR( rtuGeometryGroupRemoveChildByIndex( geometrygroup, index ) );

00304    return RT_SUCCESS;
00305  }
00306
00307  RTU_INLINE RTresult rtuGroupRemoveChildByIndex( RTgroup group, unsigned int inde
      x )
00308  {
00309    unsigned int count;
00310    RTobject temp;
00311    RTU_CHECK_ERROR( rtGroupGetChildCount( group, &count ) );
00312    RTU_CHECK_ERROR( rtGroupGetChild( group, count-1, &temp ) );
00313    RTU_CHECK_ERROR( rtGroupSetChild( group, index, temp ) );
00314    RTU_CHECK_ERROR( rtGroupSetChildCount( group, count-1 ) );
00315    return RT_SUCCESS;
00316  }
00317
00318  RTU_INLINE RTresult rtuSelectorRemoveChildByIndex( RTselector selector, unsigned
       int index )
00319  {
00320    unsigned int count;
00321    RTobject temp;
00322    RTU_CHECK_ERROR( rtSelectorGetChildCount( selector, &count ) );
00323    RTU_CHECK_ERROR( rtSelectorGetChild( selector, count-1, &temp ) );
00324    RTU_CHECK_ERROR( rtSelectorSetChild( selector, index, temp ) );
00325    RTU_CHECK_ERROR( rtSelectorSetChildCount( selector, count-1 ) );
00326    return RT_SUCCESS;
00327  }
00328
00329  RTU_INLINE RTresult rtuGeometryGroupRemoveChildByIndex( RTgeometrygroup geometry
```

```
      group, unsigned int index )
00330  {
00331    unsigned int count;
00332    RTgeometryinstance temp;
00333    RTU_CHECK_ERROR( rtGeometryGroupGetChildCount( geometrygroup, &count ) );
00334    RTU_CHECK_ERROR( rtGeometryGroupGetChild( geometrygroup, count-1, &temp ) );
00335    RTU_CHECK_ERROR( rtGeometryGroupSetChild( geometrygroup, index, temp ) );
00336    RTU_CHECK_ERROR( rtGeometryGroupSetChildCount( geometrygroup, count-1 ) );
00337    return RT_SUCCESS;
00338  }
00339
00340  RTU_INLINE RTresult rtuGroupGetChildIndex(RTgroup group, RTobject child, unsigne
      d int* index)
00341  {
00342    unsigned int count;
00343    RTobject temp;
00344    RTU_CHECK_ERROR( rtGroupGetChildCount( group, &count ) );
00345    for( *index=0; *index<count; (*index)++ ) {
00346      RTU_CHECK_ERROR( rtGroupGetChild( group, *index, &temp ) );
00347      if( child==temp ) return RT_SUCCESS;
00348    }
00349    *index = ~0u;
00350    return RT_ERROR_INVALID_VALUE;
00351  }
00352
00353  RTU_INLINE RTresult rtuSelectorGetChildIndex( RTselector selector, RTobject chil
      d, unsigned int* index )
00354  {
00355    unsigned int count;
00356    RTobject temp;
00357    RTU_CHECK_ERROR( rtSelectorGetChildCount( selector, &count ) );
00358    for( *index=0; *index<count; (*index)++ ) {
00359      RTU_CHECK_ERROR( rtSelectorGetChild( selector, *index, &temp ) );
00360      if( child==temp ) return RT_SUCCESS;
00361    }
00362    *index = ~0u;
00363    return RT_ERROR_INVALID_VALUE;
00364  }
00365
00366  RTU_INLINE RTresult rtuGeometryGroupGetChildIndex( RTgeometrygroup geometrygroup
      , RTgeometryinstance child, unsigned int* index )
00367  {
00368    unsigned int count;
00369    RTgeometryinstance temp;
00370    RTU_CHECK_ERROR( rtGeometryGroupGetChildCount( geometrygroup, &count ) );
00371    for( *index=0; *index<count; (*index)++ ) {
00372      RTU_CHECK_ERROR( rtGeometryGroupGetChild( geometrygroup, *index, &temp ) );
00373      if( child==temp ) return RT_SUCCESS;
00374    }
00375    *index = ~0u;
00376    return RT_ERROR_INVALID_VALUE;
00377  }
00378
00379 #undef RTU_CHECK_ERROR
00380 #undef RTU_INLINE
00381
00382 #endif /* __optix_optixu_h__ */
```

3.5 optixu_traversal.h File Reference

A simple API for performing raytracing queries using OptiX or the CPU.

`#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 113 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_INITOPTION_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 86 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 104 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 93 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.

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 46 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 55 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 67 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

→ ***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

→ ***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

→ ***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

→ ***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

→ ***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

→ ***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

```
00001
00002
00003 /*************************************************************************\
00004  *
00005  * Traversal API
00006  *
00007 \*************************************************************************/
00008
00023 #ifndef _optixu_optux_traversal_h_
00024 #define _optixu_optux_traversal_h_
00025
00026 #include <optix.h>
00027
00028 #ifdef __cplusplus
00029 extern "C" {
00030 #endif
00031
00035   typedef struct {
00036     int   prim_id;
00037     float t;
00038   } RTUtraversalresult;
00039
00040
00046   typedef enum {
00047     RTU_QUERY_TYPE_ANY_HIT = 0,
00048     RTU_QUERY_TYPE_CLOSEST_HIT,
00049     RTU_QUERY_TYPE_COUNT
00050   } RTUquerytype;
00051
00055   typedef enum  {
00056     RTU_RAYFORMAT_ORIGIN_DIRECTION_TMIN_TMAX_INTERLEAVED = 0,
00057     RTU_RAYFORMAT_ORIGIN_DIRECTION_INTERLEAVED,
00058     RTU_RAYFORMAT_COUNT
00059   } RTUrayformat;
00060
00067   typedef enum  {
00068     RTU_TRIFORMAT_MESH= 0,
00069     RTU_TRIFORMAT_TRIANGLE_SOUP,
00070     RTU_TRIFORMAT_COUNT
00071   } RTUtriformat;
00072
00086   typedef enum  {
00087     RTU_INITOPTION_NONE          = 0,
00088     RTU_INITOPTION_GPU_ONLY      = 1 << 0,
00089     RTU_INITOPTION_CPU_ONLY      = 1 << 1,
00090     RTU_INITOPTION_CULL_BACKFACE = 1 << 2
00091   } RTUinitoptions;
00092
00093   typedef enum  {
00094     RTU_OUTPUT_NONE        = 0,
00095     RTU_OUTPUT_NORMAL      = 1 << 0, /*< float3 [x, y, z]                      */
00096     RTU_OUTPUT_BARYCENTRIC = 1 << 1, /*< float2 [alpha, beta] (gamma implicit) */
00097     RTU_OUTPUT_BACKFACING  = 1 << 2  /*< char   [1 | 0]                       */
00098   } RTUoutput;
00099
00104   typedef enum  {
00105     RTU_OPTION_INT_NUM_THREADS=0
00106   } RTUoption;
00107
00108
00113   typedef struct RTUtraversal_api*  RTUtraversal;
00114
```

```
00115
00130   RTresult RTAPI rtuTraversalCreate( RTUtraversal* traversal,
00131                                      RTUquerytype  query_type,
00132                                      RTUrayformat  ray_format,
00133                                      RTUtriformat  tri_format,
00134                                      unsigned int  outputs,
00135                                      unsigned int  options,
00136                                      RTcontext     context );
00137
00147   RTresult RTAPI rtuTraversalGetErrorString( RTUtraversal traversal,
00148                                              RTresult code,
00149                                              const char** return_string);
00158   RTresult RTAPI rtuTraversalSetOption( RTUtraversal traversal,
00159                                         RTUoption    option,
00160                                         void*        value );
00161
00177   RTresult RTAPI rtuTraversalSetMesh( RTUtraversal    traversal,
00178                                       unsigned int    num_verts,
00179                                       const float*    verts,
00180                                       unsigned int    num_tris,
00181                                       const unsigned* indices );
00182
00197   RTresult RTAPI rtuTraversalSetTriangles( RTUtraversal traversal,
00198                                            unsigned int num_tris,
00199                                            const float* tris );
00200
00209   RTresult RTAPI rtuTraversalSetAccelData( RTUtraversal traversal,
00210                                            const void*  data,
00211                                            RTsize       data_size );
00212
00220   RTresult RTAPI rtuTraversalGetAccelDataSize( RTUtraversal traversal,
00221                                                RTsize*      data_size );
00222
00231   RTresult RTAPI rtuTraversalGetAccelData( RTUtraversal traversal,
00232                                            void*        data );
00233
00246   RTresult RTAPI rtuTraversalMapRays( RTUtraversal traversal,
00247                                       unsigned int num_rays,
00248                                       float** rays );
00249
00253   RTresult RTAPI rtuTraversalUnmapRays( RTUtraversal traversal );
00254
00262   RTresult RTAPI rtuTraversalPreprocess( RTUtraversal traversal );
00263
00270   RTresult RTAPI rtuTraversalTraverse( RTUtraversal traversal );
00271
00282   RTresult RTAPI rtuTraversalMapResults( RTUtraversal         traversal,
00283                                          RTUtraversalresult** results );
00284
00288   RTresult RTAPI rtuTraversalUnmapResults( RTUtraversal         traversal );
00289
00306   RTresult RTAPI rtuTraversalMapOutput( RTUtraversal traversal,
00307                                         RTUoutput    which,
00308                                         void**       output );
00312   RTresult RTAPI rtuTraversalUnmapOutput( RTUtraversal traversal,
00313                                           RTUoutput    which );
00321   RTresult RTAPI rtuTraversalDestroy( RTUtraversal traversal );
00322
00323 #ifdef __cplusplus
00324 } /* extern "C" */
00325 #endif
00326
00327 #endif /* _optixu_optux_traversal.h */
00328
```

Index